Benutzerhandbuch

Copyright © 1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, StyleVision, Authentic, MapForce and ALTOVA are trademarks (registered in numerous countries) of Altova GmbH.

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc., STLport ©1999,2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>. All other names or trademarks are the property of their respective owners.

ALTOVA

Altova XMLSpy 2005 Enterprise Edition

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2005

© 2005 Altova GmbH

Printed in the United States of America

5 4 3 2 1

ISBN 1-933210-08-7

Inhaltsverzeichnis

Willkommen bei der XMLSpy 2005 Enterprise Edition 1

Benutzerhandbuch 4

1 Einführung 5

1.1 Die XML-Produkte von Altova 6
1.2 Die wichtigsten Funktionen von XMLSpy 7
1.3 Benutzeroberfläche 11
1.3.1 Projektfenster 12
1.3.2 Info-Fenster 13
1.3.3 Hauptfenster 13
1.3.4 Eingabehilfen 14
1.3.5 Menü- und Symbolleiste 15

2 Tutorials 16

2.1 XMLSpy Tutorial 17
2.1.1 Die XMLSpy-Benutzeroberfläche 17
2.1.2 Erstellen eines einfachen XML-Schemas 18
– Anlegen einer neuen XML-Schema-Datei 19
– Definieren von Namespaces 22
– Definieren eines Content Model 23
– Hinzufügen von Elementen mit Drag & Drop 27
– Konfigurieren der Content Model-Ansicht 28
– Fertigstellen des Schemas 30
2.1.3 Komplexere XML-Schema-Definitionen 33
– Arbeiten mit ComplexTypes und SimpleTypes 34
– Referenzieren von globalen Elementen 41
– Attribute und Attribut-Enumerationen 43
2.1.4 Schemanavigation und Dokumentation 45
– Schemanavigation 45
– Schema-Dokumentation 47
2.1.5 Erstellen von XML Dokumenten 50

– Erstellen eines neuen XML-Dokuments ... 51
– Festlegen des Typs eines Elements ... 53
– Eingabe von Daten in der Grid-Ansicht ... 55
– Eingabe von Daten in der Textansicht ... 55
– Validierung des Dokuments ... 59
– Anhängen von Elementen und Attributen in der Grid-Ansicht ... 63
– Editieren in der Datenbank/Tabellen-Ansicht ... 64
– Ändern des Schemas ... 68
2.1.6 Transformieren von XML mit XSLT ... 70
– Zuweisen einer XSL-Datei ... 71
– Transformieren der XSL-Datei ... 72
– Ändern der XSL-Datei ... 73
2.1.7 Arbeiten mit Datenbanken ... 74
– Exportieren von XML-Daten in externe Datenbanken ... 74
– Importieren von Datenbankeninformationen ... 77
– Erstellen eines Datenbank-Schemas ... 80
Konvertieren einer Datenbank in ein Schema ... 81
2.1.8 Projektverwaltung ... 85
– Vorteile von Projekten ... 85
– Erstellen eines Projekts ... 86
2.1.9 Fertig! ... 87
2.2 Tutorial Authentic-Ansicht ... 88
2.2.1 Öffnen eines XML-Dokuments in der Authentic-Ansicht ... 88
2.2.2 Eingeben von Daten in der Authentic-Ansicht ... 89
2.2.3 Hinzufügen von Dokumentinhalt und Elementen ... 90
2.2.4 Einfügen eines Elements ... 93
2.2.5 Anwenden oder Löschen von Elementen ... 94
2.2.6 Eingabe von Attributwerten ... 94
2.2.7 Hinzufügen von Entities ... 95
2.2.8 Drucken des Dokuments ... 96
2.3 WSDL Tutorial ... 98
2.3.1 Erstellen eines Service und Port ... 98
2.3.2 Hinzufügen von Input- und Output-Messages und Parametern ... 100
2.3.3 Validieren und Fertigstellung ... 104
2.3.4 Herstellen einer Verbindung zu einem Webservice und Kopieren der WSDL-Datei ... 106
2.3.5 Senden eines SOAP Request von der WSDL-Datei ... 109
2.3.6 Erstellen von WSDL-Dokumentation ... 110

3 Textansicht 112

3.1 Visuelle Editierhilfen in der Textansicht ... 113
3.2 Eingabehilfen in der Textansicht ... 114
3.3 Bearbeiten von XML-Dokumenten ... 116
3.4 Bearbeiten von XQuery-Dokumenten ... 119

3.4.1 Öffnen eines XQuery-Dokuments ... 119
3.4.2 XQuery Eingabehilfen ... 120
3.4.3 XQuery Syntaxfärbung ... 121
3.4.4 Intelligentes Editieren von XQuery ... 123
3.4.5 Validierung und Ausführung von XQuery-Dokumenten ... 124

4 Enhanced Grid-Ansicht 126

4.1 Editieren in der Grid-Ansicht ... 130
4.2 Datenbank/Tabellen-Ansicht ... 132

5 Schema/WSDL Design-Ansicht 136

5.1 Schema Design-Ansicht ... 137
5.1.1 Schema-Übersicht ... 142
5.1.2 Content Model-Ansicht ... 146
5.1.3 Arbeiten mit SchemaAgent ... 155
– Installation von SchemaAgent Server ... 157
– Installation von SchemaAgent Client ... 158
– Herstellen einer Verbindung mit SchemaAgent Server ... 159
– Öffnen von Schemas aus dem Suchpfad ... 161
– Verwenden von Schema-Konstrukten ... 162
– Anzeigen von Schemas in SchemaAgent ... 166
– Erweiterte Validierung ... 166
5.2 WSDL Design-Ansicht ... 169
5.2.1 WSDL Design-Komponenten ... 171

6 Authentic-Ansicht 173

6.1 Übersicht über die Benutzeroberfläche ... 175
6.2 Authentic-Ansicht Symbolleistenschaltflächen ... 177
6.3 Authentic-Ansicht Hauptfenster ... 180
6.4 Authentic-Ansicht Eingabehilfen ... 183
6.5 Authentic-Ansicht Kontextmenüs ... 187
6.6 Features und ihre Verwendung ... 188
6.6.1 Tabellen in der Authentic-Ansicht ... 188
– SPS-Tabellen ... 188
– XML-Tabellen ... 189
– Symbole zum Bearbeiten von XML-Tabellen ... 193
6.6.2 Bearbeiten einer DB ... 195
– Navigieren in einer DB-Tabelle ... 195
– DB-Abfragen ... 196
– Ändern einer DB-Tabelle ... 200
6.6.3 Datumswahl ... 201

6.6.4 Definieren von Entities 202
6.6.5 Grafiken in der Authentic-Ansicht 204
6.6.6 Verwendung von Tasten in der Authentic-Ansicht 204

7 Browser-Ansicht 205

8 XMLSpy 2005 in MS Visual Studio .NET 206

8.1 Unterschiede zwischen der .NET und der Standalone-Version 207
8.2 Visual Studio .NET und XSLT Debugger 211
8.3 Visual Studio .NET und SOAP Debugger 214

9 XMLSpy 2005 in Eclipse 216

9.1 Starten von Eclipse und Verwendung des XMLSpy 2005 Plug-Ins 217
9.2 Erstellen von XML-Dateien 219
9.3 Importieren von XML-Dateien in Eclipse-Ordner 221
9.4 Unterschiede zwischen Eclipse und den Standalone-Versionen 224
9.5 Eclipse-Plattform und XSLT Debugger 225
9.6 Eclipse Plattform und SOAP Debugger 227
9.7 Eclipse-Ansichten und -Perspektiven 230

10 XSLT und XQuery Debugger 232

10.1 Funktionsweise und Benutzeroberfläche 233
10.2 Befehle und Symbolleistenschaltflächen 235
10.3 Einstellungen 238
10.4 Starten einer Debugger-Sitzung 240
10.5 Informationsfenster 242
10.5.1 Kontext 243
10.5.2 Variablen 243
10.5.3 XPath-Watch 244
10.5.4 Call Stack 244
10.5.5 Messages 245
10.5.6 Templates 245
10.5.7 Info 246
10.5.8 Anordnen der Informationsfenster 246
10.6 Breakpoints 248

11 SOAP Debugger 251

11.1 SOAP-Kommunikationsvorgang 252

11.2 Verwendung des SOAP Debuggers 253
11.3 Setzen von Breakpoints im SOAP Debugger 259

12 Code Generator 262

12.1 Einführung in Code Generator 263
12.2 Generieren von Quellcode anhand eines Schemas 265
12.3 Beispiele für die Verwendung des generierten Java-, C++ und C#-Codes 268
12.3.1 Erzeugen von XML-Dateien 268
12.3.2 Öffnen und Parsen vorhandener XML-Dateien 271
12.4 Der Weg zu SPL (Spy Programming Language) 274
12.4.1 Code-Blöcke 274
12.4.2 Kommentare 274
12.4.3 Variablen 275
12.4.4 Globale Objekte 275
12.4.5 Verwendung von Dateien 275
12.4.6 Bedingungen 276
12.4.7 foreach 277
12.4.8 Hinzufügen von Text zu Dateien 277
12.4.9 Subroutinen 277
– Deklaration einer Subroutine 278
– Subroutinenaufruf 278
– Beispiel für eine Subroutine 279
12.4.10 Integrierte Typen 280
– Namespace 280
– Klasse 280
– Member 282
– Facet 283
– Enumeration 283
12.5 Fehlercodes 284

13 Benutzerreferenz 285

13.1 Menü "Datei" 286
13.1.1 Neu... 286
13.1.2 Öffnen 289
13.1.3 URL öffnen 289
13.1.4 Neu laden 291
13.1.5 Kodierung 291
13.1.6 Schließen 292
13.1.7 Alle schließen 292
13.1.8 Speichern 292
13.1.9 Speichern unter 292
13.1.10 Speichern unter URL 292

13.1.11 Alles speichern 294
13.1.12 Als Mail senden 294
13.1.13 Drucken 295
13.1.14 Druckansicht 296
13.1.15 Druckereinrichtung 296
13.1.16 Zuletzt verwendete Dateien 297
13.1.17 Beenden 297
13.2 Menü "Bearbeiten" 298
13.2.1 Rückgängig 298
13.2.2 Wiederherstellen 298
13.2.3 Ausschneiden 298
13.2.4 Kopieren 298
13.2.5 Einfügen 299
13.2.6 Löschen 299
13.2.7 Als XML-Text kopieren 299
13.2.8 Als strukturierten Text kopieren 299
13.2.9 XPath kopieren 301
13.2.10 XML-Text Druckaufbereitung 301
13.2.11 Alles markieren 302
13.2.12 Suchen 302
13.2.13 Weitersuchen 305
13.2.14 Ersetzen 305
13.2.15 Lesezeichen einfügen/löschen 307
13.2.16 Alle Lesezeichen löschen 307
13.2.17 Zum nächsten Lesezeichen 308
13.2.18 Zum vorhergehenden Lesezeichen 308
13.3 Menü "Projekt" 309
13.3.1 Neu 310
13.3.2 Öffnen 310
13.3.3 Neu laden 310
13.3.4 Schließen 310
13.3.5 Speichern 310
13.3.6 Versionskontrolle 310
– Öffnen 312
– Versionskontrolle aktivieren 314
– Aktuellste Version holen 315
– Auschecken 315
– Einchecken 316
– Auschecken rückgängig 316
– Zu Versionskontrolle hinzufügen 317
– Von Versionskontrolle ausgliedern 319
– Verlauf anzeigen 319
– Unterschiede anzeigen 320
– Eigenschaften 321
– Status aktualisieren 322

– Native Oberfläche ausführen ... 322
13.3.7 Dateien hinzufügen ... 322
13.3.8 URL hinzufügen ... 322
13.3.9 Aktive Datei hinzufügen ... 323
13.3.10 Aktive und verwandte Dateien hinzufügen ... 323
13.3.11 Projektordner in Projekt einfügen ... 323
13.3.12 Externes Verzeichnis zum Projekt hinzufügen ... 323
13.3.13 Externes Webverzeichnis zum Projekt hinzufügen ... 326
13.3.14 Projekteigenschaften ... 328
13.3.15 Zuletzt verwendete Projekte ... 330
13.4 Menü "XML" ... 331
13.4.1 Einfügen ... 331
– Attribut einfügen ... 331
– Element einfügen ... 331
– Text einfügen ... 331
– CDATA einfügen ... 332
– Comment einfügen ... 332
– XML einfügen ... 332
– Processing Instruction einfügen ... 332
– DOCTYPE einfügen ... 332
– ExternalID einfügen ... 333
– ELEMENT einfügen ... 334
– ATTLIST einfügen ... 334
– ENTITY einfügen ... 334
– NOTATION einfügen ... 334
13.4.2 Anhängen ... 334
– Attribut anhängen ... 335
– Element anhängen ... 335
– Text anhängen ... 335
– CDATA anhängen ... 336
– Comment anhängen ... 336
– XML anhängen ... 336
– Processing Instruction anhängen ... 336
– DOCTYPE anhängen ... 336
– ExternalID anhängen ... 337
– ELEMENT anhängen ... 338
– ATTLIST anhängen ... 338
– ENTITY anhängen ... 338
– NOTATION anhängen ... 338
13.4.3 Child hinzufügen ... 338
– Child-Attribut hinzufügen ... 339
– Child-Element hinzufügen ... 339
– Child-Text hinzufügen ... 339
– Child CDATA hinzufügen ... 340
– Child Comment hinzufügen ... 340

– Child XML hinzufügen ... 340
– Child Processsing Instruction hinzufügen ... 340
– Child DOCTYPE hinzufügen ... 340
– Child ExternalID hinzufügen ... 340
– Child ELEMENT hinzufügen ... 341
– Child ATTLIST hinzufügen ... 341
– Child ENTITY hinzufügen ... 341
– Child NOTATION hinzufügen ... 341
13.4.4 Konvertieren in ... 342
– In Attribut konvertieren ... 342
– In Element konvertieren ... 342
– In Text konvertieren ... 342
– In CDATA konvertieren ... 342
– In Comment konvertieren ... 343
– In XML konvertieren ... 343
– In Processing Instruction konvertieren ... 343
– In DOCTYPE konvertieren ... 343
– In ExternalID konvertieren ... 343
– In ELEMENT konvertieren ... 343
– In ATTLIST konvertieren ... 343
– In ENTITY konvertieren ... 343
– In NOTATION konvertieren ... 343
13.4.5 Tabelle ... 344
– Als Tabelle anzeigen ... 344
– Zeile einfügen ... 345
– Zeile anhängen ... 345
– Aufsteigend sortieren ... 345
– Absteigend sortieren ... 345
13.4.6 Höher stufen ... 346
13.4.7 Tiefer stufen ... 346
13.4.8 In Element einbinden ... 346
13.4.9 XPath evaluieren ... 347
13.4.10 Wohlgeformtheit prüfen ... 350
13.4.11 Validieren ... 351
13.4.12 Eingabehilfen aktualisieren ... 352
13.4.13 Namespace Präfix ... 352
13.5 Menü "DTD/Schema" ... 354
13.5.1 DTD zuweisen ... 354
13.5.2 Schema zuweisen ... 354
13.5.3 Weitere DTD einfügen ... 354
13.5.4 Gehe zu DTD ... 355
13.5.5 Gehe zu Schema ... 355
13.5.6 Gehe zu Definition ... 355
13.5.7 DTD/Schema generieren ... 355
13.5.8 DTD/Schema konvertieren ... 356

13.5.9 In MapForce auf andere(s) DTD/Schema oder DB mappen ... 359
13.5.10 HTML/PDF-Ausgabe in StyleVision erstellen ... 359
13.5.11 XML Beispieldatei erzeugen ... 359
13.5.12 Programmcode generieren ... 360
13.5.13 Cache-Speicher leeren ... 360
13.6 Menü "Schema-Design" ... 361
13.6.1 Schema-Einstellungen ... 361
13.6.2 Diagramm speichern ... 362
13.6.3 Schema-Dokumentation generieren ... 362
13.6.4 Ansicht konfigurieren ... 364
13.6.5 Vergrößern/Verkleinern ... 367
13.6.6 Globale Elemente anzeigen ... 368
13.6.7 Diagramm anzeigen ... 368
13.6.8 Oracle Schema Extensions aktivieren ... 368
13.6.9 Oracle Schema-Einstellungen ... 369
13.6.10 Microsoft SQL Server Schema Extensions aktivieren ... 369
13.6.11 Named Schema Relationships ... 370
13.6.12 Unnamed Element Relationships ... 371
13.6.13 Tamino Schema Extensions aktivieren ... 371
13.6.14 Tamino Schema-Eigenschaften ... 372
13.6.15 Verbindung zu SchemaAgent Server herstellen ... 373
13.6.16 Verbindung mit SchemaAgent Server trennen ... 373
13.6.17 In SchemaAgent anzeigen ... 373
13.6.18 Erweiterte Validierung ... 373
13.7 Menü "XSL/XQuery" ... 374
13.7.1 XSL-Transformation ... 374
13.7.2 XSL:FO-Transformation ... 374
13.7.3 XSL-Parameter/XQuery-Variablen ... 376
13.7.4 XQuery-Ausführung ... 379
13.7.5 XSL zuweisen ... 379
13.7.6 XSL:FO zuweisen ... 380
13.7.7 XML-Beispieldatei zuweisen ... 380
13.7.8 Gehe zu XSL ... 380
13.7.9 Debugger starten ... 380
13.7.10 Debugger anhalten ... 381
13.7.11 Debugger neu starten ... 381
13.7.12 Debugger-Sitzung beenden ... 381
13.7.13 Einsteigen ... 381
13.7.14 Aussteigen ... 381
13.7.15 Überspringen ... 382
13.7.16 Aktuellen Execution Node einblenden ... 382
13.7.17 Breakpoint einfügen/löschen ... 382
13.7.18 Breakpoint ein/aus ... 382
13.7.19 Breakpoints ... 382
13.7.20 Debug-Fenster ... 383

13.7.21 XSLT/XQuery-Einstellungen ... 383
13.8 Menü "Authentic" ... 384
13.8.1 Neues Dokument ... 384
13.8.2 Datenbankdaten bearbeiten ... 385
13.8.3 StyleVision Stylesheet zuweisen ... 385
13.8.4 StyleVision Stylesheet bearbeiten ... 386
13.8.5 XML Entities definieren ... 386
13.8.6 Markupsymbole ausblenden ... 388
13.8.7 Kleine Markup-Symbole einblenden ... 388
13.8.8 Große Markup-Symbole einblenden ... 388
13.8.9 Gemischte Markup-Symbole einblenden ... 388
13.8.10 Zeile anhängen ... 388
13.8.11 Zeile einfügen ... 388
13.8.12 Zeile kopieren ... 388
13.8.13 Zeile nach oben ... 389
13.8.14 Zeile nach unten ... 389
13.8.15 Zeile löschen ... 389
13.9 Menü "Konvertieren" ... 390
13.9.1 Textdatei importieren ... 390
13.9.2 Datenbank importieren ... 391
– MS Access als Quelle ... 392
– MS SQL als Quelle ... 394
ADO Connection String ... 395
– Oracle als Quelle ... 397
– IBM DB2 als Quelle ... 398
– Datenbankimport-Einstellungen ... 401
13.9.3 Microsoft Word Dokument importieren ... 402
13.9.4 XML-Schema von DB-Struktur erzeugen ... 403
– XML-Schema von DB-Struktur erzeugen (MS Access) ... 404
– XML-Schema von DB-Struktur erzeugen (Microsoft SQL) ... 407
– XML-Schema von DB-Struktur erzeugen (Oracle) ... 410
– XML-Schema von DB-Struktur erzeugen (IBM DB2) ... 412
– Optionen zum Erzeugen von XML-Schemas ... 415
– Schema-Extensions erzeugen ... 417
– Mappen auf XML-Schema-Datentypen ... 418
XML-Datentypen und MS SQL Server ... 418
XML-Datentypen und MS Access ... 419
XML-Datentypen und Oracle ... 421
XML-Datentypen und MySQL ... 422
XML-Datentypen und Sybase ... 424
XML-Datentypen und ODBC ... 425
XML-Datentypen und ADO ... 427
13.9.5 DB-Import basierend auf XML-Schema ... 428
13.9.6 DB-Struktur von XML-Schema erzeugen ... 432
– Mappen von XML-Schema-Datentypen ... 439

XML-Datentypen und MS SQL Server ... 439
XML-Datentypen und MS Access ... 441
XML-Datentypen und Oracle ... 443
XML-Datentypen und MySQL ... 445
13.9.7 Export in Textdatei/Datenbank ... 448
– Export in Datenbank ... 449
– Export in Textdatei ... 450
13.9.8 Tamino ... 451
– Installation von Tamino ... 451
– Installation von WebDAV ... 451
– Tamino Schema-Extensions ... 452
– Erstellen, Speichern und Auflisten von Tamino Schemas ... 452
Legacy Schemas in Tamino hinzufügen ... 458
Einschränkungen für Tamino Schema Definitionen ... 459
– Bearbeiten von Tamino Schemas ... 460
– Erstellen von XML-Dateien, die auf Tamino Schemas basieren ... 460
– Abfrage von Daten in Tamino ... 462
13.9.9 Oracle XML DB ... 466
– Suchen... ... 466
– Schemas auflisten ... 467
– Schema hinzufügen ... 468
– Oracle XML-Dokumente durchsuchen ... 470
– Eigenschaften ... 470
– XML Dokumentinstanzen hinzufügen ... 471
13.10 Menü "Ansicht" ... 473
13.10.1 Text-Ansicht ... 473
13.10.2 Enhanced Grid Ansicht ... 473
13.10.3 Schema/ WSDL Design-Ansicht ... 474
13.10.4 Authentic-Ansicht ... 474
13.10.5 Browser-Ansicht ... 474
13.10.6 Erweitern ... 474
13.10.7 Reduzieren ... 475
13.10.8 Alles erweitern ... 475
13.10.9 Unselektierte reduzieren ... 475
13.10.10 Optimale Breite ... 475
13.10.11 Zeilenumbruch ... 475
13.10.12 Gehe zu Zeile/Zeichen ... 476
13.10.13 Gehe zu Datei ... 476
13.10.14 Zeilennummerierungsleiste ... 476
13.10.15 Lesezeichenleiste ... 477
13.10.16 Klappleiste ... 477
13.10.17 Einrücklinien ... 477
13.11 Menü "Browser" ... 478
13.11.1 Zurück ... 478
13.11.2 Vorwärts ... 478

13.11.3 Abbrechen ... 478
13.11.4 Aktualisieren ... 478
13.11.5 Schriftgröße ... 478
13.11.6 Neues Fenster ... 478
13.12 Menü "WSDL" ... 480
13.12.1 Messages ... 480
13.12.2 Operations ... 480
13.12.3 PortType ... 480
13.12.4 Binding ... 481
13.12.5 Service ... 481
13.12.6 Types ... 481
13.12.7 Schema-Dokumentation generieren ... 482
13.12.8 Einstellungen ... 482
13.13 Menü "Soap" ... 483
13.13.1 SOAP Request erzeugen ... 483
13.13.2 SOAP Request an Server senden ... 484
13.13.3 SOAP Request-Parameter ändern ... 486
13.13.4 Soap Debugger-Sitzung ... 487
13.13.5 Starten ... 487
13.13.6 Einzelschritt ... 488
13.13.7 Beim nächsten Request anhalten ... 488
13.13.8 Bei nächstem Response anhalten ... 488
13.13.9 Proxyserver anhalten ... 488
13.13.10 Soap Debugger Einstellungen ... 488
13.14 Menü "Extras" ... 490
13.14.1 Rechtschreibung ... 490
13.14.2 Rechtschreibung - Optionen ... 492
13.14.3 Skripting-Umgebung ... 496
13.14.4 Makros ... 496
13.14.5 Projekt ... 496
– Skripts zuweisen ... 497
– Projekt-Skripts entkoppeln ... 497
– Projekt-Skripts aktivieren ... 497
13.14.6 Vergleiche ... 497
– Geöffnete Datei vergleichen mit ... 497
– Verzeichnisse vergleichen ... 499
– Vergleichseinstellungen ... 501
13.14.7 Anpassen ... 504
– Befehle ... 504
– Symbolleisten ... 505
– Tastatur ... 507
– Menü ... 511
– Makros ... 513
– Plug-Ins ... 514
– Optionen ... 516

– Kontextmenü 'Anpassen' 516
13.14.8 Optionen 518
– Datei 518
– Dateiarten 520
– Bearbeiten 522
– Ansicht 523
– Grid-Schriftarten 524
– Schemaschriftarten 524
– Textschriftarten 525
– Farben 526
– Kodierung 528
– XSL 529
– Skripting-Umgebung 530
13.15 Menü "Fenster" 532
13.15.1 Überlappend 532
13.15.2 Horizontal anordnen 532
13.15.3 Vertikal anordnen 532
13.15.4 Projekt-Fenster 532
13.15.5 Info-Fenster 532
13.15.6 Eingabehilfen 532
13.15.7 Alle ein/aus 533
13.15.8 Liste aller offenen Fenster 533
13.16 Menü "Hilfe" 534
13.16.1 Inhaltsverzeichnis... 534
13.16.2 Index 534
13.16.3 Suche... 534
13.16.4 Tastaturzuordnungen 534
13.16.5 Registrieren 535
13.16.6 Bestellformular... 536
13.16.7 Support Center 536
13.16.8 Fragen und Antworten im Web 536
13.16.9 Komponenten downloaden... 536
13.16.10 XMLSpy im Internet 537
13.16.11 XMLSpy Training 537
13.16.12 Über XMLSpy... 537
13.17 Altova XSLT 1.0-Prozessor 538
13.18 Altova XSLT 2.0-Prozessor 539
13.19 Altova XQuery 1.0-Prozessor 542

Referenz für Programmierer 546

1 Release Notes 547

2 Scripting 548

2.1 Overview 549
2.1.1 Scripting projects....549
2.1.2 Running Forms, Event Handlers, and Macros....551
2.1.3 Scripting Settings....551
2.2 The Scripting Environment GUI 553
2.2.1 Project Window....553
2.2.2 Main Window....555
2.2.3 Form Object Bar....556
2.2.4 The Form Editor....557
2.3 Using the Scripting Environment 562
2.3.1 Creating an Event Handler....562
2.3.2 Creating a Form....564
2.3.3 Writing a Macro....569
2.3.4 Calling macros....571
2.3.5 Programming points to note....573
2.4 Menus 574
2.4.1 File 574
2.4.2 Edit 574
2.4.3 View 575
2.4.4 Project 575
2.4.5 Layout 576
2.4.6 Draw 579
2.4.7 Window....580

3 XMLSpy Plugins 581

3.1 Registration of IDE PlugIns 582
3.2 Configuration XML 583
3.3 ATL sample files 586
3.3.1 Interface description (IDL)....586
3.3.2 Class definition....588
3.3.3 Implementation....588
3.4 IXMLSpyPlugIn 591
3.4.1 OnCommand....591
3.4.2 OnUpdateCommand....592
3.4.3 OnEvent....592
3.4.4 GetUIModifications....594
3.4.5 GetDescription....595

4 The XMLSpy API 596

4.1 Overview 597
- 4.1.1 Object model 597
- 4.1.2 Simple document access 598
- 4.1.3 Error handling 599
- 4.1.4 Events 601
- 4.1.5 Import and export of data 604
- 4.1.6 Using XMLData to modify document structure 607
- 4.1.7 The DOM and XMLData 610
- 4.1.8 Obsolete Authentic Row operations 612
- 4.1.9 Obsolete Authentic Editing operations 613

4.2 Usage Examples 614
- 4.2.1 JScript: Bubble Sort Dynamic Tables 614
- 4.2.2 VBScript: Using object-level events 615

4.3 Interfaces 617
- 4.3.1 Application 617
  - – Events 618
    - *OnBeforeOpenDocument* 618
    - *OnBeforeOpenProject* 619
    - *OnDocumentOpened* 619
    - *OnProjectOpened* 620
  - – ActiveDocument 620
  - – AddMacroMenuItem 621
  - – Application 621
  - – ClearMacroMenu 621
  - – CurrentProject 621
  - – Dialogs 622
  - – Documents 622
  - – GetDatabaseImportElementList 622
  - – GetDatabaseSettings 623
  - – GetDatabaseTables 623
  - – GetExportSettings 624
  - – GetTextImportElementList 624
  - – GetTextImportExportSettings 625
  - – ImportFromDatabase 625
  - – ImportFromSchema 626
  - – ImportFromText 627
  - – ImportFromWord 628
  - – NewProject 628
  - – OpenProject 628
  - – Parent 629
  - – Quit 629
  - – ShowApplication 629

– ShowForm ... 630
– URLDelete ... 630
– URLMakeDirectory ... 630
– Visible ... 631
– WarningNumber ... 631
– WarningText ... 631
4.3.2 AuthenticDataTransfer ... 632
– dropEffect ... 632
– getData ... 633
– ownDrag ... 633
– type ... 633
4.3.3 AuthenticRange ... 633
– AppendRow ... 636
– Application ... 636
– CanPerformAction ... 636
– CanPerformActionWith ... 637
– Clone ... 637
– CollapsToBegin ... 638
– CollapsToEnd ... 638
– Copy ... 638
– Cut ... 638
– Delete ... 639
– DeleteRow ... 639
– DuplicateRow ... 640
– ExpandTo ... 640
– FirstTextPosition ... 640
– FirstXMLData ... 641
– FirstXMLDataOffset ... 642
– GetElementAttributeNames ... 643
– GetElementAttributeValue ... 643
– GetElementHierarchy ... 644
– GetEntityNames ... 644
– Goto ... 645
– GotoNext ... 645
– GotoNextCursorPosition ... 646
– GotoPrevious ... 646
– GotoPreviousCursorPosition ... 647
– HasElementAttribute ... 647
– InsertEntity ... 647
– InsertRow ... 648
– IsCopyEnabled ... 648
– IsCutEnabled ... 649
– IsDeleteEnabled ... 649
– IsEmpty ... 649
– IsEqual ... 649

– IsFirstRow 650
– IsInDynamicTable 650
– IsLastRow 650
– IsPasteEnabled 651
– IsTextStateApplied 651
– LastTextPosition 651
– LastXMLData 652
– LastXMLDataOffset 653
– MoveBegin 654
– MoveEnd 654
– MoveRowDown 655
– MoveRowUp 655
– Parent 655
– Paste 656
– PerformAction 656
– Select 657
– SelectNext 657
– SelectPrevious 658
– SetElementAttributeValue 659
– SetFromRange 660
– Text 660
4.3.4 AuthenticView 660
– Events 662
OnBeforeCopy 662
OnBeforeCut 662
OnBeforeDelete 663
OnBeforeDrop 663
OnBeforePaste 664
OnDragOver 665
OnKeyboardEvent 666
OnMouseEvent 667
OnSelectionChanged 668
– Application 669
– AsXMLString 669
– DocumentBegin 670
– DocumentEnd 670
– Event 670
– Goto 670
– IsRedoEnabled 671
– IsUndoEnabled 671
– MarkupVisibility 672
– Parent 672
– Print 672
– Redo 673
– Selection 673

– Undo ... 673
– UpdateXMLInstanceEntities ... 674
– WholeDocument ... 674
– XMLDataRoot ... 675
4.3.5 CodeGeneratorDlg ... 675
– Application ... 676
– CPPSettings_DOMType ... 676
– CPPSettings_LibraryType ... 676
– CPPSettings_UseMFC ... 676
– CSharpSettings_ProjectType ... 677
– OutputPath ... 677
– OutputPathDialogAction ... 677
– OutputResultDialogAction ... 677
– Parent ... 678
– ProgrammingLanguage ... 678
– PropertySheetDialogAction ... 678
– TemplateFileName ... 678
4.3.6 DatabaseConnection ... 679
– ADOConnection ... 679
– AsAttributes ... 680
– CreateMissingTables ... 680
– CreateNew ... 680
– DatabaseKind ... 680
– ExcludeKeys ... 681
– File ... 681
– IncludeEmptyElements ... 681
– NumberDateTimeFormat ... 681
– ODBCConnection ... 682
– SQLSelect ... 682
– TextFieldLen ... 682
4.3.7 Dialogs ... 682
– Application ... 683
– CodeGeneratorDlg ... 683
– FileSelectionDlg ... 683
– Parent ... 684
– SchemaDocumentationDlg ... 684
4.3.8 Document ... 684
– Events ... 686
OnBeforeSaveDocument ... 686
OnBeforeCloseDocument ... 686
OnCloseDocument ... 687
OnViewActivation ... 687
– Application ... 688
– AssignDTD ... 688
– AssignSchema ... 688

– AssignXSL ... 689
– AssignXSLFO ... 689
– AuthenticView ... 689
– Close ... 690
– ConvertDTDOrSchema ... 690
– CreateChild ... 691
– CreateSchemaDiagram ... 691
– CurrentViewMode ... 691
– DataRoot ... 692
– DocEditView ... 692
– Encoding ... 692
– EndChanges ... 693
– ExportToDatabase ... 693
– ExportToText ... 694
– ExecuteXQuery ... 695
– FullName ... 695
– GenerateDTDOrSchema ... 695
– GenerateProgramCode ... 696
– GenerateSchemaDocumentation ... 696
– GetExportElementList ... 697
– GetPathName (obsolete) ... 697
– GridView ... 697
– IsModified ... 698
– IsValid ... 698
– IsWellFormed ... 699
– Name ... 700
– Parent ... 700
– Path ... 700
– RootElement ... 700
– Save ... 701
– SaveAs ... 701
– Saved ... 701
– SaveInString ... 702
– SaveToURL ... 702
– SetActiveDocument ... 702
– SetEncoding (obsolete) ... 703
– SetPathName (obsolete) ... 704
– StartChanges ... 704
– SwitchViewMode ... 704
– Title ... 705
– TransformXSL ... 705
– TransformXSLFO ... 705
– UpdateViews ... 705

4.3.9 Documents ... 706

– Count ... 706

– Item ... 707
– NewFile ... 707
– NewFileFromText ... 707
– OpenFile ... 707
– OpenURL ... 708
– OpenURLDialog ... 708
4.3.10 ElementList ... 709
– Count ... 709
– Item ... 710
– RemoveElement ... 710
4.3.11 ElementListItem ... 710
– ElementKind ... 710
– FieldCount ... 710
– Name ... 711
– RecordCount ... 711
4.3.12 ExportSettings ... 711
– CreateKeys ... 711
– ElementList ... 712
– EntitiesToText ... 712
– ExportAllElements ... 712
– FromAttributes ... 712
– FromSingleSubElements ... 712
– FromTextValues ... 713
– IndependentPrimaryKey ... 713
– Namespace ... 713
– SubLevelLimit ... 713
4.3.13 FileSelectionDlg ... 713
– Application ... 714
– DialogAction ... 714
– FullName ... 714
– Parent ... 715
4.3.14 GridView ... 715
– Events ... 715
OnBeforeDrag ... 715
OnBeforeDrop ... 716
OnBeforeStartEditing ... 716
OnEditingFinished ... 717
OnFocusChanged ... 717
– CurrentFocus ... 717
– Deselect ... 718
– IsVisible ... 718
– Select ... 718
– SetFocus ... 718
4.3.15 SchemaDocumentationDlg ... 718
– Application ... 719

– IncludeEverything ... 720
– AllDetails ... 720
– IncludeAttributeGroups ... 720
– IncludeComplexTypes ... 720
– IncludeGlobalElements ... 721
– IncludeGroups ... 721
– IncludeIndex ... 721
– IncludeLocalElements ... 722
– IncludeRedefines ... 722
– IncludeSimpleTypes ... 722
– OptionsDialogAction ... 722
– OutputFile ... 723
– OutputFileDialogAction ... 723
– OutputFormat ... 723
– Parent ... 724
– ShowAnnotations ... 724
– ShowAttributes ... 724
– ShowChildren ... 725
– ShowDiagram ... 725
– ShowEnumerations ... 725
– ShowIdentityConstraints ... 725
– ShowNamespace ... 726
– ShowPatterns ... 726
– ShowProgressBar ... 726
– ShowProperties ... 727
– ShowResult ... 727
– ShowSingleFacets ... 727
– ShowSourceCode ... 728
– ShowType ... 728
– ShowUsedBy ... 728

4.3.16 SpyProject ... 729
– CloseProject ... 729
– ProjectFile ... 729
– RootItems ... 729
– SaveProject ... 730
– SaveProjectAs ... 730

4.3.17 SpyProjectItem ... 730
– ChildItems ... 730
– FileExtensions ... 731
– ItemType ... 731
– Name ... 731
– Open ... 731
– ParentItem ... 731
– Path ... 732
– ValidateWith ... 732

– XMLForXSLTransformation ... 732
– XSLForXMLTransformation ... 732
– XSLTransformationFileExtension ... 732
– XSLTransformationFolder ... 732
4.3.18 SpyProjectItems ... 733
– AddFile ... 733
– AddFolder ... 733
– AddURL ... 733
– Count ... 734
– Item ... 734
– RemoveItem ... 734
4.3.19 TextImportExportSettings ... 734
– DestinationFolder ... 735
– EnclosingCharacter ... 735
– Encoding ... 735
– EncodingByteOrder ... 735
– FieldDelimiter ... 736
– FileExtension ... 736
– HeaderRow ... 736
– ImportFile ... 736
4.3.20 XMLData ... 736
– AppendChild ... 737
– EraseAllChildren ... 738
– EraseCurrentChild ... 738
– GetCurrentChild ... 739
– GetFirstChild ... 739
– GetNextChild ... 740
– HasChildren ... 741
– InsertChild ... 741
– IsSameNode ... 742
– Kind ... 742
– MayHaveChildren ... 742
– Name ... 742
– Parent ... 743
– TextValue ... 743
4.4 Interfaces (obsolete) ... 744
4.4.1 AuthenticEvent (obsolete) ... 744
– altKey (obsolete) ... 745
– altLeft (obsolete) ... 746
– button (obsolete) ... 747
– cancelBubble (obsolete) ... 748
– clientX (obsolete) ... 749
– clientY (obsolete) ... 750
– ctrlKey (obsolete) ... 751
– ctrlLeft (obsolete) ... 752

– dataTransfer (obsolete) 753
– fromElement (obsolete) 754
– keyCode (obsolete) 754
– propertyName (obsolete) 755
– repeat (obsolete) 755
– returnValue (obsolete) 755
– shiftKey (obsolete) 756
– shiftLeft (obsolete) 757
– srcElement (obsolete) 758
– type (obsolete) 759
4.4.2 AuthenticSelection (obsolete) 760
– End (obsolete) 761
– EndTextPosition (obsolete) 761
– Start (obsolete) 762
– StartTextPosition (obsolete) 762
4.4.3 OldAuthentictView (obsolete) 763
– ApplyTextState (obsolete) 765
– CurrentSelection (obsolete) 766
– EditClear (obsolete) 766
– EditCopy (obsolete) 767
– EditCut (obsolete) 767
– EditPaste (obsolete) 768
– EditRedo (obsolete) 768
– EditSelectAll (obsolete) 769
– EditUndo (obsolete) 769
– event (obsolete) 770
– GetAllowedElements (obsolete) 770
– GetNextVisible (obsolete) 773
– GetPreviousVisible (obsolete) 774
– IsEditClearEnabled (obsolete) 775
– IsEditCopyEnabled (obsolete) 775
– IsEditCutEnabled (obsolete) 776
– IsEditPasteEnabled (obsolete) 776
– IsEditRedoEnabled (obsolete) 777
– IsEditUndoEnabled (obsolete) 777
– IsRowAppendEnabled (obsolete) 778
– IsRowDeleteEnabled (obsolete) 778
– IsRowDuplicateEnabled (obsolete) 779
– IsRowInsertEnabled (obsolete) 779
– IsRowMoveDownEnabled (obsolete) 780
– IsRowMoveUpEnabled (obsolete) 780
– IsTextStateApplied (obsolete) 781
– IsTextStateEnabled (obsolete) 781
– LoadXML (obsolete) 782
– MarkUpView (obsolete) 782

– RowAppend (obsolete) 783
– RowDelete (obsolete) 783
– RowDuplicate (obsolete) 784
– RowInsert (obsolete) 784
– RowMoveDown (obsolete) 785
– RowMoveUp (obsolete) 785
– SaveXML (obsolete) 786
– SelectionMoveTabOrder (obsolete) 787
– SelectionSet (obsolete) 788
– XMLRoot (obsolete) 789
4.5 Enumerations 791
4.5.1 SPYAuthenticActions 791
4.5.2 SPYAuthenticDocumentPosition 791
4.5.3 SPYAuthenticElementActions 791
4.5.4 SPYAuthenticElementKind 792
4.5.5 SPYAuthenticMarkupVisibility 792
4.5.6 SPYDatabaseKind 792
4.5.7 SPYDialogAction 793
4.5.8 SPYDOMType 793
4.5.9 SPYDTDSchemaFormat 793
4.5.10 SPYEncodingByteOrder 793
4.5.11 SPYExportNamespace 793
4.5.12 SPYFrequentElements 794
4.5.13 SPYKeyEvent 794
4.5.14 SPYLibType 794
4.5.15 SPYLoading 794
4.5.16 SPYMouseEvent 794
4.5.17 SPYNumberDateTimeFormat 795
4.5.18 SPYProgrammingLanguage 795
4.5.19 SPYProjectItemTypes 796
4.5.20 SPYProjectType 796
4.5.21 SPYSchemaDefKind 796
4.5.22 SPYSchemaDocumentationFormat 797
4.5.23 SPYTextDelimiters 797
4.5.24 SPYTextEnclosing 798
4.5.25 SPYTypeDetection 798
4.5.26 SPYURLTypes 798
4.5.27 SPYViewModes 798
4.5.28 SPYVirtualKeyMask 799
4.5.29 SPYXMLDataKind 799

5 Using the XMLSpy API with Java 801

5.1 Sample source code 803
5.2 SpyApplication 805

5.3 SpyCodeGeneratorDlg 806
5.4 SpyDatabaseConnection 807
5.5 SpyDialogs 808
5.6 SpyDoc 809
5.7 SpyDocuments 811
5.8 SpyElementList 812
5.9 SpyElementListItem 813
5.10 SpyExportSettings 814
5.11 SpyFileSelectionDlg 815
5.12 SpyGridView 816
5.13 SpyProject 817
5.14 SpyProjectItem 818
5.15 SpyProjectItems 819
5.16 SpySchemaDocumentationDlg 820
5.17 SpyTextImportExportSettings 822
5.18 SpyXMLData 823
5.19 Authentic 824
5.19.1 SpyAuthenticRange 824
5.19.2 SpyAuthenticView 825
5.19.3 SpyDocEditSelection 825
5.19.4 SpyDocEditView 825
5.20 Predefined constants 827
5.20.1 SPYAuthenticActions 827
5.20.2 SPYAuthenticDocumentPosition 827
5.20.3 SPYAuthenticElementKind 827
5.20.4 SPYAuthenticMarkupVisibility 828
5.20.5 SPYDatabaseKind 828
5.20.6 SPYDialogAction 828
5.20.7 SPYDOMType 828
5.20.8 SPYDTDSchemaFormat 829
5.20.9 SPYEncodingByteOrder 829
5.20.10 SPYExportNamespace 829
5.20.11 SPYFrequentElements 829
5.20.12 SPYLibType 829
5.20.13 SPYLoading 830
5.20.14 SPYNameDateTimeFormat 830
5.20.15 SPYProgrammingLanguage 830
5.20.16 SPYProjectItemTypes 830
5.20.17 SPYProjectType 830
5.20.18 SPYSchemaDefKind 831
5.20.19 SPYSchemaDocumentationFormat 831
5.20.20 SPYTextDelimiters 831
5.20.21 SPYTextEnclosing 832
5.20.22 SPYTypeDetection 832
5.20.23 SPYURLTypes 832

5.20.24 SpyViewModes....832
5.20.25 SPYWhitespaceComparison....833
5.20.26 SPYXMLDataKind....833

Anhänge 836

1 Technische Hintergrundinformationen 837

1.1 XML Parser....838
1.2 Schema-Dialekte....839
1.3 Altova XSLT- und XQuery-Prozessor....840
1.4 OS- und Arbeitsspeicheranforderungen....841
1.5 Internet-Verwendung....842
1.6 Unicode-Unterstützung....843
1.6.1 Windows NT4.0/2000/XP....843
1.6.2 Rechts nach Links-Schreibrichtung....844
1.7 Lizenzüberwachung....845

2 Lizenzierung und Vertrieb 846

2.1 Electronic Software Distribution (EDS)....847
2.2 Copyright....848
2.3 Software-Produktlizenz....849

Index 859

Altova XMLSpy 2005 Enterprise Edition

Willkommen bei der XMLSpy 2005 Enterprise Edition

Willkommen bei der XMLSpy 2005 Enterprise Edition

Altova XMLSpy® 2005 ist die Industriestandard XML-Entwicklungsumgebung zum Erstellen, Editieren und Debuggen von Unternehmensapplikationen, die auf XML, XML-Schema, XSLT, XQuery, SOAP, WSDL und Webservice-Technologien basieren. Mit Hilfe dieser Anwendung können J2EE-, .NET- und Datenbank-Entwickler nun schneller und effizienter arbeiten.

Copyright © 1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. Altova, XMLSpy, MapForce, StyleVision, SchemaAgent, Authentic and DiffDog are trademarks (registered in numerous countries) of Altova GmbH. Patent(s) pending.

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc., STLport ©1999,2000 Boris Fomitchev, © 1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>. All other names or trademarks are the property of their respective owners.

ALTOVA®

Altova XMLSpy 2005 Enterprise Edition

Benutzerhandbuch

Benutzerhandbuch

Dieses Benutzerhandbuch enthält Tutorials und Erklärungen zu den verschiedenen Features von XMLSpy 2005, um Ihnen eine Einführung in das Programm zu geben. Außerdem finden Sie darin ein umfassendes Referenzhandbuch, in dem die Funktionen der XMLSpy 2005 IDE beschrieben sind. Das Benutzerhandbuch ist in die folgenden Abschnitte gegliedert:

- eine Einführung, die eine Übersicht über die XML-Produkte von Altova enthält.
- ein **Tutorial,** in dem beschrieben wird, wie XMLSpy 2005 für die verschiedenen Aspekte von XML verwendet wird:
  - XML-Editierung und -Validierung
  - Schema/DTD-Editierung und -Validierung
  - XSLT-Editierung und -Transformation
- ausführliche Beschreibungen der fünf Ansichten von XMLSpy 2005: Textansicht, Enhanced Grid-Ansicht, Schema/WSDL-Ansicht, Authentic-Ansicht und Browser-Ansicht. In diesen fünf Abschnitten werden die GUI und Features der verschiedenen Ansichten sowie die verschiedenen Komponenten und Symbole der einzelnen Ansichten beschrieben und Sie erfahren, wie man mit den Funktionen dieser Ansichten arbeitet.
- eine Beschreibung der Integration von XMLSpy 2005 mit Visual Studio .NET.
- eine Beschreibung der Verwendung der **XSLT/XQuery Debugger**-Komponente, die in Ihrem XMLSpy 2005 Produkt integriert ist. Mit Hilfe des XSLT/XQuery Debuggers können Sie Ihre XSLT 1.0 und XSLT 2.0 Stylesheets und XQuery-Dokumente überprüfen und korrigieren. In einer XSLT Debugger-Sitzung können Sie die Transformation einer XML-Datei anhand eines ausgewählten Stylesheet Schritt für Schritt durchgehen. In einer XQuery Debugger-Sitzung können Sie ein XQuery-Dokument Schritt für Schritt durchgehen und eventuelle Fehler ausfindig machen.
- eine Beschreibung des **SOAP Debuggers** von XMLSpy 2005.
- eine Beschreibung der **Codegenerierungsfunktionen** von XMLSpy 2005.
- eine **Benutzerreferenz**, die eine Beschreibung aller Menübefehle von XMLSpy 2005 enthält.

© 2005 Altova GmbH

1 Einführung

Diese Einführung enthält einen kurzen Überblick über:

- die verschiedenen XML-Produkte von Altova und wie sie einander ergänzen
- die wichtigsten Funktionen von XMLSpy
- die Benutzeroberfläche

Dieser Abschnitt soll Ihnen einen allgemeinen Überblick über XMLSpy 2005 verschaffen und Sie mit den Funktionen des Produkts und der Benutzeroberfläche vertraut machen.

1.1 Die XML-Produkte von Altova

Die XLM-Produkte von Altova sind einfach zu bedienen und decken alle XML-Anforderungen ab. Sie ergänzen einander und bieten Ihnen eine umfassende XML-Entwicklungsumgebung. Derzeit umfasst die Altova-Produktpalette die folgenden Produkte.

Altova XMLSpy® 2005 ist eine umfassende IDE zur Entwicklung von XML-Projekten und steht in drei Versionen zur Verfügung. Den größten Funktionsumfang bietet die Enterprise Edition mit Funktionen zum Erzeugen und Bearbeiten von DTDs, XML-Schemas, XML-Dateien und XSLT Stylesheets. Es stehen Ihnen leistungsstarke Editierfeatures, mehrere Dokumentenansichten - einschließlich der Altova Authentic-Ansicht - Funktionen zum Validieren und für XSLT-Transformationen mittels eines integrierten Prozessors zur Verfügung. Des Weiteren bietet das Programm Import- und Exportfunktionen für Textdateien und Datenbanken, Funktionen zur Verwaltung von XML-Dokumenten in Projekten, einen XSLT- und XQuery Debugger, einen WSDL Editor sowie Funktionen zur Codegenerierung. Die Professional Edition und die Home Edition enthalten einen anders konfigurierten Funktionsumfang.

Altova StyleVision® 2005 ist eine neue Lösung zum Erstellen komplexer XSLT Stylesheets mit Hilfe einer intuitiven Drag-and-Drop-Oberfläche. Mit StyleVision lassen sich auch Authentic Stylesheets erstellen, über die Sie die Anzeige und Dateneingabe in XML-Dokumente in der Authentic-Ansicht steuern können. StyleVision 2005 ist in Form einer Enterprise und einer Professional Edition erhältlich.

Altova Authentic® 2005 (Desktop und Browser Edition) Diese beiden Versionen sind einem Textverarbeitungsprogramm ähnliche Editierprogramme, die die Eingabe von Daten in XML-Dokumente mittels Formularen unterstützen. Sie können damit Komponenten wie Grafiken und Tabellen einfügen und Dateien in Echtzeit gegen ein Schema validieren.

Altova MapForce™ 2005 dient zum Mappen von einem Schema auf ein anderes und zum Konvertieren von schemabasierten XML-Dateien in XML-Dateien, die auf einem anderen Schema basieren.

Altova Website
Werfen Sie bitte auch gelegentlich einen Blick auf die Altova Website, www.altova.com, wo Sie sich über Produktneuheiten informieren können sowie Updates, zusätzliche Dokumentation und Support erhalten.

Beachten Sie bitte, dass die Benutzerhandbücher für alle Altova-Produkte in den folgenden Formaten zur Verfügung stehen:

- Online-Handbücher: stehen auf der Support-Seite der Altova Website zur Verfügung
- PDFs zum Ausdrucken, die Sie von der Altova Website herunterladen und ausdrucken können
- Druckausgabe der Handbücher, die Sie über einen Link auf der Altova Website bestellen und erwerben können.

Die Dokumentation auf der Website wird regelmäßig aktualisiert und entspricht dem Stand der aktuellen Version.

Support und Feedback
Wenn Sie weitere Informationen benötigen oder Fragen zu unseren Produkten haben, besuchen Sie unser Support Center auf der Altova Website. Hier finden Sie:

- Links zu unseren FAQ-Seiten
- Diskussionsforen zu Altova-Produkten und allgemeinen XML-Themen
- Online Support-Formulare, die an unser Support-Team gesendet werden.

Wir freuen uns auch über Feedback zu unserer Dokumentation.

© 2005 Altova GmbH

1.2 Die wichtigsten Funktionen von XMLSpy

XMLSpy 2005 ist eine IDE (Integrated Development Environment) zur Entwicklung von XML-Projekten. Mit XMLSpy 2005 können Sie unter anderem die verschiedensten XML- und Textdokumente editieren und verarbeiten; Daten in XML-Dokumente importieren und aus diesen exportieren (sowie Daten in Datenbanken importieren bzw. aus diesen exportieren); Konvertierungen zwischen bestimmten Arten von XML-Dokumenten und anderen Dokumentarten vornehmen; unterschiedliche Arten von XML-Dokumenten in Projekten verknüpfen; Dokumente mittels des integrierten XSLT 1.0-, XSLT 2.0- und XQuery 1.0-Prozessors verarbeiten, und sogar Code anhand von XML-Dokumenten generieren.

XMLSpy 2005 bietet außerdem eine grafische Bearbeitungsansicht für XML-Dokumente, nämlich die beliebte Altova Authentic-Ansicht, über die Benutzer Daten ähnlich wie bei einem Textverarbeitungsprogramm in ein XML-Dokument eingeben können. Die Authentic-Ansicht eignet sich vor allem für folgende Zwecke:

- wenn Mitarbeiter ohne XML-Kenntnisse Daten in XML-Dokumente eingeben müssen
- wenn mehrere Benutzer Daten in ein einziges Dokument eingeben wollen/auf dieses zugreifen wollen, das über einen Server oder eine gemeinsam benutzte Ressource zugänglich ist.

Dieser Abschnitt enthält einen kurzen Überblick über die **wichtigsten Funktionen** von XMLSpy 2005, die dann in den verschiedenen Abschnitten zu den Ansichten der Benutzeroberfläche (Textansicht, Schema/WSDL-Ansicht, Authentic-Ansicht usw.) und in der Benutzerreferenz näher beschrieben sind. Beachten Sie bitte, dass es sich hierbei nicht um eine komplette Liste aller Features handelt. In diesem Abschnitt sollen Sie nur einen ungefähren Eindruck davon erhalten, wofür Sie XMLSpy 2005 verwenden können.

Bearbeiten von XML-Dokumenten in mehreren Editierformaten
Sie können ein XML-Dokument als reinen Text (Textansicht), in einem hierarchischen Tabellenformat (Enhanced Grid-Ansicht) oder in einer grafischen WYSIWYG-Ansicht (Authentic-Ansicht) bearbeiten. Für XML-Schemas und WSDL-Dokumente können Sie auch die Schema/WSDL-Ansicht verwenden, eine grafische Benutzeroberfläche, die das Erstellen von komplexen Schemas und WSDL-Dokumenten viel einfacher macht. Je nach Bedarf können Sie zwischen den verschiedenen Ansichten hin- und herwechseln. Über die Browser-Ansicht können Sie XML-Dokumente, die mit einem XSLT Stylesheet und HTML-Dokumenten verknüpft sind, direkt anzeigen lassen.

Wohlgeformtheitsprüfung und integrierter Validator
Alle XML-Dokumente werden auf ihre Wohlgeformtheit überprüft, wenn Sie die Ansicht wechseln oder die Datei speichern. Außerdem können XML-Dokumente validiert werden, wenn ein Schema (DTD oder XML-Schema) mit dem XML-Dokument verknüpft wurde. Andere Dokumentarten wie z.B. DTDs werden außerdem auch auf Fehler in der Syntax und Struktur überprüft.

Strukturelle Editierfunktionen
In der Textansicht helfen Ihnen Features wie z.B. Zeilennummerierung, Einrücklinien, Lesezeichen und ein- und ausklappbare Elemente schnell und effizient, durch das Dokument zu navigieren.

Intelligentes Editieren
Wenn ein Schema mit einem XML-Dokument verknüpft wurde, stellt die Autokomplettierungsfunktion der Textansicht eine nützliche Editierhilfe dar. Während der

Eingabe erscheinen an der Cursorposition Popup-Menüs, die die Elemente, Attribute und Attributwerte enthalten, die an dieser Stelle eingefügt werden dürfen. Außerdem werden beim Einfügen von Anfangs-Tags automatisch die richtigen End-Tags eingefügt. Attribute, die Sie aus den Popup-Menüs auswählen, werden mit öffnenden und schließenden Anführungszeichen eingefügt. XMLSpy 2005 kann außerdem so konfiguriert werden, dass obligatorische Elemente und/oder Attribute beim Einfügen eines Elements automatisch eingefügt werden. Des Weiteren hat jede Ansicht einen Satz von Eingabehilfen, über die Sie Dokumentenkomponenten einfügen oder die Eigenschaften der ausgewählten Komponente im Hauptfenster festlegen können.

Bearbeiten und Verwalten von Schemas
In der grafischen Schema/WSDL-Ansicht lassen sich Schemas schnell und einfach erstellen, da hier keine eingehenden Kenntnisse der Struktur, der Syntax und der Grundlagen des Erstellens von XML-Schemas nötig sind. Sie können auch DTDs erstellen, die auf die korrekte Syntax überprüft werden, und Schemas in DTDs und umgekehrt konvertieren sowie Dokumentation generieren. Zusätzlich dazu haben Sie mit Hilfe des SchemaAgent die Möglichkeit, Schemas von anderen Servern zu verwenden, wodurch Ihnen eine effiziente XML-Umgebung zum Verwalten und Bearbeiten von XML-Schemas zur Verfügung steht.

Integrierter XSLT 1.0- und XSLT 2.0-Prozessor
Die integrierten XSLT 1.0- und XSLT 2.0-Prozessoren entsprechen den Vorgaben der jeweiligen W3C Drafts. Sie gestatten Ihnen das Transformieren von XML-Dokumenten mit Hilfe eines XSLT 1.0- oder XSLT 2.0-Stylesheet direkt von der IDE aus sowie das Debuggen von XSLT Stylesheets mit dem XSLT Debugger.

Integrierter XQuery 1.0-Prozessor
Der integrierte XQuery 1.0-Prozessor entspricht den Standards des XQuery 1.0 W3C Working Draft vom 23. Juli 2004. Sie können damit XQuery-Dokumente direkt von der IDE aus ausführen und debuggen.

Transformation von XML-Dokumenten
XML-Dokumente können direkt in der IDE transformiert werden, entweder mit dem internen (integrierten) XSLT-Prozessor oder jedem beliebigen externen XSLT-Prozessor. Um eine PDF-Datei von der IDE aus zu generieren, können Sie einen externen FO-Prozessor definieren und auch XML nach Angabe des zu verwendenden XSLT Stylesheet mit einem einzigen Klick ins PDF-Format transformieren. Außerdem können von der IDE selbst Parameterwerte an die XSLT-Transformation übergeben werden.

XPath-Evaluierung
Die Funktion "XPath evaluieren" listet für das gewünschte Dokument die Sequence (oder das Node Set) auf, die/das von einem XPath-Ausdruck zurückgegeben wird. Als Context Node können Sie entweder den Dokumenten-Node oder das ausgewählte Element auswählen. Die Funktion "XPath evaluieren" eignet sich vor allem dann, wenn Sie ein XSLT Stylesheet erstellen und einen XPath-Ausdruck dafür evaluieren möchten. Sie können mit Hilfe der Node-Liste in der zurückgegebenen Sequence zu den einzelnen Nodes navigieren.

XSLT Debugger für XSLT 1.0 und XSLT 2.0
Der XSLT Debugger für XSLT 1.0 und XSLT 2.0 entspricht den Standards der entsprechenden W3C Drafts. Mit Hilfe des XSLT Debuggers kann ein XSLT Stylesheet debuggt werden. Der Debugger führt das zu debuggende XSLT Stylesheet an der XML-Datei aus. Das Resultat wird Schritt für Schritt für jeden einzelnen Schritt der Transformation generiert, wobei Sie den Context Node, die gerade ausgeführte Vorlage und andere Einzelheiten jedes einzelnen

 © 2005 Altova GmbH

Transformationsschritts sehen.

XQuery 1.0 Debugger
Der XQuery 1.0 Debugger entspricht den Standards des XQuery 1.0 W3C Working Draft vom 23. Juli 2004. Er ist dem XSLT Debugger ähnlich und kann zum Debuggen von XQuery-Dokumenten verwendet werden.

XML-Projektverwaltung
Die XMLSpy 2005 IDE gestattet Ihnen, miteinander verwandte Dateien in Form einer Baumstruktur in Projekten zu organisieren. Ein Projekt kann aus Schema-Dateien, XML-Dateien, Transformationsdateien und Ausgabedateien bestehen. Die Dateien eines Projekts werden gemeinsam im Projektfenster aufgelistet, sodass Sie bei der Arbeit an einem Projekt in übersichtlicher Form jederzeit zur Verfügung stehen. Zudem können Sie bestimmte Einstellungen für das gesamte Projekt oder für einzelne Ordner festlegen, wie z.B. die Schema-Datei, gegen die Dateien validiert werden sollen oder die zu verwendende Transformationsdatei.

Authentic-Ansicht
Die Authentic-Ansicht ist eine grafische Ansicht des XML-Dokuments in XMLSpy 2005. Über diese Ansicht, können Daten wie über ein Textverarbeitungsprogramm in das XML-Dokument eingegeben werden. Das StyleVision Power Stylesheet, welches in Altova StyleVision 2005 erstellt wird, dient dazu, die Formatierung des XML-Dokuments in der Authentic-Ansicht zu definieren und festzulegen, wie die Dateneingabe in Altova StyleVision 2005 erfolgt. Beachten Sie bitte, dass die Authentic-Ansicht auch in der derzeit kostenlos erhältlichen Authentic 2005 Desktop Edition von Altova zur Verfügung steht.

Datenbankimport
Sie können Datenbankdaten als XML-Datei importieren und anhand der Datenbankstruktur eine XML-Schema-Datei generieren. Es wird der Import der folgenden Datenbanken unterstützt: MS Access, MS SQL Server, Oracle, MySQL, Sybase, IBM DB2.

WSDL und SOAP
Die Schema/WSDL-Ansicht ermöglicht Ihnen das Erstellen und Bearbeiten von WSDL-Dokumenten über eine leicht verständliche grafische Benutzeroberfläche. Auch SOAP Requests lassen sich über die IDE erstellen und debuggen.

Vergleich von XML-Dateien (Differencing)
Mit Hilfe der Vergleichsfunktion in XMLSpy 2005 können Sie Unterschiede zwischen zwei XML-Dateien ausfindig machen. Dabei lassen sich die verschiedensten Optionen einstellen, um den Vergleich zu konfigurieren, wie z.B. ob die Reihenfolge von Attributen oder Child-Elementen ignoriert werden soll, ob Entities aufgelöst oder Namespaces ignoriert werden sollen. Mit Hilfe der Vergleichsfunktion lassen sich auch Ordner vergleichen.

Visual Studio .NET Integration
XMLSpy 2005 lässt sich in Ihre Visual Studio .NET-Umgebung integrieren. Sie müssen dazu nur eine EXE-Datei von der Altova Website herunterladen und ausführen.

Eclipse 3.0 Integration
Eclipse 3.0 ist eine Open Source-Umgebung, in der verschiedene Arten von Applikationen in Form von Plug-Ins integriert werden können. XMLSpy 2005 für die Eclipse-Plattform ist ein

Eclipse 3.0 Plug-In, mit Hilfe dessen Sie die Funktionen einer zuvor installierten XMLSpy 2005 Version direkt von der Eclipse 3.0-Plattform aus nutzen können.

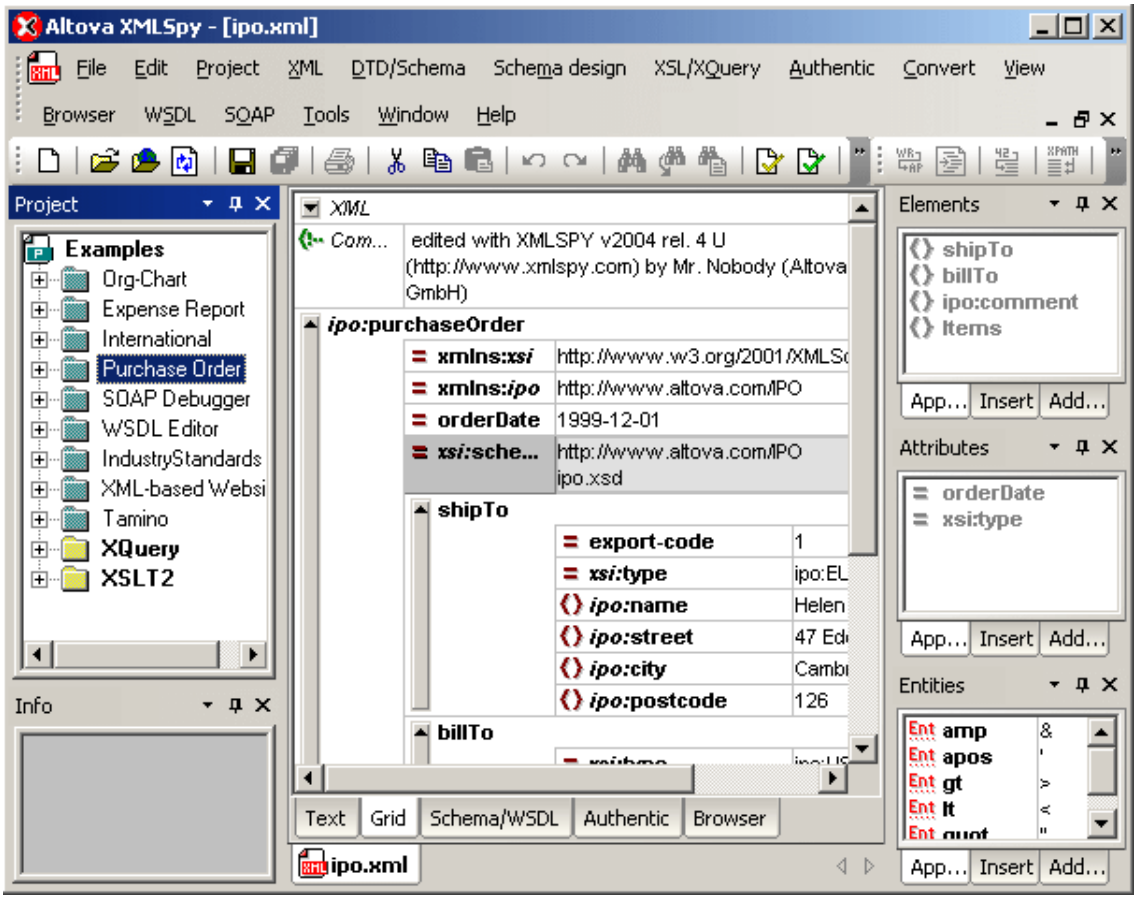
Codegenerierung
Wenn Sie Java-, C++ oder C#-Code verwenden, um Daten in XML-Dateien zu manipulieren, könnten Sie Code mit Klassendefinitionen für das Schema (DTD oder XML-Schema) generieren, auf dem das XML-Dokument basiert. In XMLSpy 2005 lässt sich derartiger Code direkt anhand von DTDs und XML-Schemas generieren.

© 2005 Altova GmbH

1.3 Benutzeroberfläche

XMLSpy 2005 hat eine grafische Benutzeroberfläche (siehe Abb. unten), die in drei Bereiche aufgegliedert ist:

- das **Projektfenster**, in dem Sie Dateien und Dateigruppen in Projekten organisieren und editieren können; und das **Info-Fenster**, in dem Metainformationen über das derzeit bearbeitete Dokument angezeigt werden.
- das **Hauptfenster**, in dem das Dokument, das Sie gerade bearbeiten, angezeigt wird. Welche Dokumentansichten im Hauptfenster zur Verfügung stehen, hängt von der Art des aktiven Dokuments ab. Sie können jederzeit zwischen den verschiedenen Ansichten wechseln.
- **Eingabehilfe-Fenster**, die je nach Art das angezeigten Dokuments und je nach der Ansicht, die im Hauptfenster ausgewählt ist, variieren können. Die Eingabehilfen dienen zum grafischen Editieren eines Dokuments.

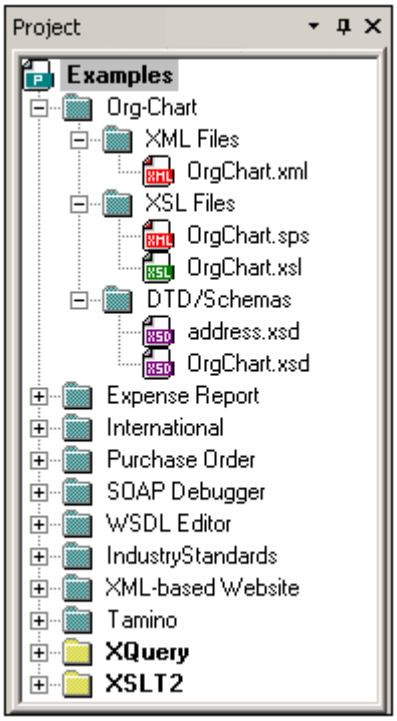


Diese Fenster können unterhalb der Menü- und Symbolleiste angedockt werden (siehe Abbildung oben) oder unterhalb der Menü- und Symbolleiste an beliebiger Stelle angeordnet werden. Position und Größe lassen sich durch Ziehen mit der Maus anpassen. Sie können auch die Befehle im Menü **Fenster** zum Ein- und Ausblenden der Eingabehilfen verwenden.

Dieser Abschnitt enthält eine Einführung in die Grundzüge der Benutzeroberfläche. Ausführliche Beschreibungen zu den verschiedenen Bereichen der Benutzeroberfläche folgen in diesem Abschnitt.

1.3.1 Projektfenster

XMLSpy 2005 verwendet zur Verwaltung mehrerer Dateien oder URLs in XML-Projekten die bekannte Baumstrukturansicht. Sie können Dateien und URLs nach Erweiterung oder beliebigen anderen Kriterien in Ordnern gruppieren, um die Struktur übersichtlicher zu gestalten und Batch-Verarbeitungen zu erleichtern.



Ordner können physischen Verzeichnissen in Ihrem Dateisystem entsprechen oder Sie können Dateierweiterungen für die einzelnen Ordner definieren, nach denen die Dateien geordnet werden. Projektordner sind "semantische" Ordner, die eine logische Dateigruppierung darstellen und **nicht** der hierarchischen Organisation von Dateien auf Ihrer Festplatte entsprechen müssen.

Zuweisen von XSL-Transformationen zu Projektordnern
Sie können jedem Ordner andere XSL-Transformationsparameter zuweisen und sogar ein und dieselbe physische Datei gleichzeitig in mehreren Projektordnern ablegen. Dies eignet sich vor allem, wenn Sie Ihre Daten in einer XML-Datei speichern wollen, aber verschiedene XSL Stylesheets verwenden, um unterschiedliche Ausgabedateien zu erzeugen (z.B. eine separate HTML- und WML-Darstellung).

Zuweisen von DTDs / Schemas zu Projektordnern
Sie können verschiedenen Ordnern unterschiedliche DTDs oder Schemas zuweisen. Auf diese Art können Sie eine Datei sowohl gegen eine DTD als auch gegen ein XML-Schema validieren, ohne die Datei selbst zu ändern, was von Vorteil ist, wenn Sie sich gerade in der Phase des Umstiegs von DTDs auf Schemas befinden.

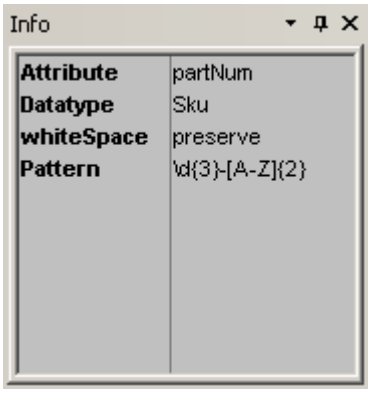
Verwenden Sie zur Projektverwaltung die Befehle im Menü Projekt.

Hinweis: Über die Menüoption **Fenster | Projekt-Fenster** können Sie das Projektfenster ein- oder ausblenden.

© 2005 Altova GmbH

1.3.2 Info-Fenster

XMLSpy 2005 enthält ein praktisches Informationsfenster, das detaillierte Informationen über das Element bzw. Attribut enthält, auf dem der Cursor gerade positioniert ist.



Diese Informationen stehen in allen Bearbeitungsansichten zur Verfügung und sind besonders in Verbindung mit der `xsd:annotation`-Funktion nützlich.

Hinweis: Sie können das Info-Fenster über die Menüoption **Fenster | Info-Fenster** ein- oder ausblenden.

1.3.3 Hauptfenster

Das Hauptfenster ist der Bereich, in dem alle Dokumente in XMLSpy 2005 angezeigt und bearbeitet werden.

Verwalten mehrerer offener Dateien

- Sie können in XMLSpy 2005 eine beliebige Anzahl an Dokumenten gleichzeitig geöffnet haben und bearbeiten.
- Jede offene Datei im Hauptfenster wird in einem eigenen Dokumentenfenster angezeigt. Jede dieser Dateien verfügt über ein Register mit dem Namen der Datei, das am unteren Rand des Hauptfensters angezeigt wird.
- Wenn offene Dateien überlappend, nebeneinander angezeigt oder minimiert werden, wird eine Titelleiste mit (i) dem Namen der Datei und (ii) den Standardschaltflächen zum Minimieren, Maximieren und Schließen angezeigt.
- Offene Dateien können maximiert oder minimiert werden, indem Sie in der Titelleiste der offenen Datei auf die jeweilige Schaltfläche zum **Maximieren** bzw. **Minimieren** klicken.
- Wenn Sie eine Datei maximieren, werden auch alle anderen offenen Dateien maximiert.
- Um offene Dateien überlappend oder nebeneinander anzuzeigen, verwenden Sie die Befehle im Menü Fenster.
- Um eine Datei zur aktiven zu machen (um sie zu bearbeiten), klicken Sie auf das Register der Datei oder an eine beliebe Stelle in der jeweiligen Datei. Alternativ dazu können Sie die gewünschte Datei auch im Menü **Fenster** aus der Liste der offenen Dateien im unteren Bereich des Menüs auswählen.
- Sie können offene Dateien auch durch Verwendung von **Strg+Tab** oder **Strg+F6** in der Reihenfolge aktivieren, in der sie geöffnet wurden.

Schneller Zugriff auf Dateibefehle
Um Dateibefehle schnell aufzurufen (wie z.B. Drucken, Schließen, als E-Mail-Anhang versenden), rechtsklicken Sie auf das Dateiregister. Dadurch wird das Kontextmenü mit einer Auswahl an Dateibefehlen geöffnet.

Ansichten im Hauptfenster
XMLSpy 2005 bietet verschiedene Ansichten für Ihr XML-Dokument. Dabei handelt es sich entweder um Editier- oder Browser-Ansichten:

- **Textansicht:** Eine Editieransicht mit Syntaxfärbung zur Bearbeitung einer Datei auf Source-Ebene
- **Enhanced Grid-Ansicht:** Zur strukturierten Bearbeitung. Das Dokument wird in Tabellenform dargestellt und kann grafisch bearbeitet werden. Diese Ansicht enthält auch eine eingebettete **Datenbank / Tabellen-Ansicht**, in der sich wiederholende Elemente in Tabellenform dargestellt werden
- **Schema/WSDL-Ansicht:** Zum Anzeigen und Bearbeiten von XML-Schemas und WSDL-Dokumenten
- **Authentic-Ansicht:** Zum Bearbeiten von XML-Dokumenten auf Basis von StyleVision Power Stylesheets
- **Browser-Ansicht:** Eine integrierte Browser-Ansicht, die sowohl CSS- als auch XSL-Stylesheets unterstützt.

Um im Dokumentenfenster zwischen Dokumentenansichten zu wechseln, klicken Sie auf die entsprechende Ansichtsschaltfläche am unteren Rand des Dokumentenfensters oder verwenden Sie die Befehle im Menü Ansicht.

Hinweis: Sie können die Standardansicht (d.h. die Ansicht im Hauptfenster) für bestimmte Dateierweiterungen anpassen. Gehen Sie dazu zu **Extras | Optionen** und nehmen Sie auf den Registern **Dateiarten** und **Ansicht**, die gewünschten Einstellungen vor.

1.3.4 Eingabehilfen

XMLSpy 2005 verfügt über intelligente Editierfeatures, mit Hilfe derer Sie schnell gültige XML-Dokumente erstellen können. Diese Features sind in drei palettenartigen Fenstern organisiert, den so genannten Eingabehilfen.

Wenn Sie ein Dokument bearbeiten, sehen Sie in den Eingabehilfen je nach Cursorposition Bearbeitungsoptionen zur Struktur. Die Eingabehilfen beziehen die erforderlichen Informationen aus der zugrundeliegenden DTD, dem XML-Schema und/oder dem StyleVision Power Stylesheet. Wenn Sie z.B. ein XML-Datendokument bearbeiten, werden in den entsprechenden Eingabehilfenfenstern die Elemente, Attribute und Entities angezeigt (sowie dazugehörige Informationen), die an der aktuellen Cursorposition eingefügt werden können.

Die Eingabehilfenfenster haben in Visual Studio .NET ein XMLSpy 2005-Präfix.

Eingabehilfen in verschiedenen Ansichten
Welche Eingabehilfen angezeigt werden, ist abhängig von der gewählten Ansicht. Die verschiedenen Gruppen von Eingabehilfen werden nach den in Ihrem Altova-Produkt verfügbaren Ansichten eingeteilt:

- Textansicht und Grid-Ansicht: Element-, Attribut- und Entities-Eingabehilfe
- Schema Design-Ansicht: Komponenten-Navigator sowie Details- und Facets-Eingabehilfe
- WSDL Design-Ansicht: Übersicht und Details-Eingabehilfe
- Authentic: Element-, Attribut- und Entities-Eingabehilfe

© 2005 Altova GmbH

Eine Beschreibung der Eingabehilfen der einzelnen Ansichten finden Sie weiter hinten in der Dokumentation im Abschnitt über die jeweilige Ansicht.

Hinweis: Über die Menüoption **Fenster | Eingabehilfen** können Sie die Eingabehilfen ein- oder ausblenden.

1.3.5 Menü- und Symbolleiste

Menüleiste
Die Menüleiste enthält die verschiedenen Menüs der Applikation. Es gelten die folgenden Konventionen:

- Wenn Befehle in einem Menü in einer Ansicht oder an einer bestimmten Stelle im Dokument **nicht** zur Verfügung stehen, sind sie ausgegraut und deaktiviert.
- Bei Auswahl einiger Menübefehle erscheint ein Untermenü mit einer Liste zusätzlicher Optionen. Solche Menübefehle weisen rechts neben dem Befehlsnamen einen nach rechts weisenden Pfeil auf.
- Bei einigen Menübefehlen wird ein Dialogfeld aufgerufen, in dem Sie weitere Informationen eingeben müssen, damit der Befehl ausgeführt werden kann. Diese Befehle sind durch drei Auslassungspunkte (...) nach dem Namen des Befehls gekennzeichnet.
- Um einen Menübefehl aufzurufen, klicken Sie auf den Menünamen und anschließend auf den Befehl. Enthält ein Menüeintrag ein Untermenü, wird dieses geöffnet, wenn Sie die Maus über den Menüeintrag positionieren. Klicken Sie auf den gewünschten Untermenüeintrag.
- Sie können ein Menü auch durch Drücken der entsprechenden Tastenkombination öffnen. Die Tastenkombination für die einzelnen Menüs lautet **Alt+TASTE**, wobei TASTE für den unterstrichenen Buchstaben im Menünamen steht. Die Tastenkombination für das Menü **Datei** ist z.B. **Alt+a**.
- Die Auswahl eines Menübefehls (also eines Befehls in einem Menü) erfolgt durch das nacheinander-Auswählen (i) des Menüs durch die entsprechende Tastenkombination (siehe voriger Punkt) und (ii) anschließendes Drücken der Tastenkombination für den jeweiligen Befehl (**Alt+TASTE**, wobei TASTE dem unterstrichenen Buchstaben im Befehlsnamen entspricht). Um z.B. eine neue Datei zu erstellen, (**Datei | Neu**), drücken Sie **Alt+a** und dann **Alt+N**.
- Einige Menübefehle können **direkt** durch Drücken eines speziellen **Tastenkürzels** oder einer Tastenkombination (**Strg+TASTE**) aufgerufen werden. Bei Befehlen, denen ein Tastenkürzel zugewiesen wurde, wird rechts vom Befehl das Tastenkürzel oder die Tastenkombination angezeigt. So können Sie z.B. eine neue Datei mit der Tastenkombination **Strg+N** erstellen und eine XML-Datei durch Drücken von **F8** validieren.

Symbolleiste
Die Symbolleiste enthält Schaltflächen zum Aufrufen von Menübefehlen. Wenn Sie den Mauszeiger über die Schaltfläche halten, wird der Name des Befehls angezeigt. Um den Befehl auszuführen, klicken Sie auf die Schaltfläche.

Die Schaltflächen der Symbolleiste sind in Gruppen angeordnet. Im Dialogfeld **Extras | Anpassen | Symbolleisten** können Sie einstellen, welche Symbolleistengruppen angezeigt werden sollen. Sie können Symbolleistengruppen auf der GUI auch an eine andere Stelle ziehen, indem Sie darauf klicken und mit dem Ziehpunkt an die gewünschte Stelle verschieben.

2 Tutorials

Dieser Abschnitt enthält die folgenden Tutorials:

- XMLSPY Tutorial: Ein Tutorial, in dem Schritt für Schritt beschrieben wird, wie ein XML-Schema erstellt wird, bzw. eine XML-Datei erstellt, bearbeitet und transformiert wird, Datenbanken in XMLSpy 2005 für XML verwendet werden und zusammengehörige Dateien in XMLSpy-Projekten organisiert werden können.
- Authentic-Ansicht Tutorial: Ein Tutorial, in dem die einzigartige Benutzeroberfläche der Altova Authentic-Ansicht und deren Features beschrieben werden. In der Authentic-Ansicht können Sie XML-Dokumente über eine graphische Benutzeroberfläche bearbeiten, ohne XML-Kenntnisse zu benötigen oder die Struktur des zugrunde liegenden Schemas zu kennen.
- WSDL Tutorial: ein Tutorial, in dem die Grundzüge der WSDL Design-Ansicht, in der Sie WSDL-Dokumente schnell und einfach erstellen können, erklärt werden.

© 2005 Altova GmbH

2.1 XMLSpy Tutorial

Mit der Ihnen vorliegenden Anleitung erhalten Sie einen umfassenden Einblick in XML. Schritt für Schritt werden Sie mit den Funktionen von XMLSpy 2005 vertraut.

Mit dieser Anleitung lernen Sie folgendes:

- Generieren von **einfachen Schemas**
- **Verallgemeinern von Schemas** durch simple und complex types
- Erstellen von Schema-**Dokumentation**
- Erstellen von **XML-Dokumenten** basierend auf einer Schema-Datei
- Kopieren von XML-Daten in andere Anwendungen (z.B. Excel), sowie Wiedereinfügen in XMLSpy 2005
- **Validierung** des XMS Dokumentes gegen das zugewiesene Schema
- **Aktualisierung** von Schema-Einstellungen während der Bearbeitung des XML Dokumentes
- **Transformieren** des XML-Dokumentes in HTML durch Verwendung von XSLT und Anzeigen der Ergebnisse in der Browser-Ansicht
- **Import** und **Export** von Datenbankeneinträgen von und nach XMLSpy 2005
- Erstellen eines **Schemas** anhand von einer MS Access-Datenbank
- Erstellen eines XMLSpy 2005-**Projektes** zum Organisieren Ihrer XML-Dokumente

Installation und Konfiguration
In dieser Anleitung gehen wir davon aus, dass Sie XMLSpy 2005 erfolgreich auf Ihrem PC installiert haben, dass Sie einen kostenlosen Evaluierungs-Key erhalten haben oder dass Sie sich als User bereits registriert haben. Die Evaluierungsversion von XMLSpy 2005 ist voll funktionsfähig. Allerdings ist diese auf dreißig Tage begrenzt. Sie können danach eine reguläre Lizenz über unseren sicheren Webserver oder über einen Vertriebspartner erwerben.

Anleitung: Beispieldateien
Beispieldateien, die in der vorliegenden Anleitung angesprochen werden, finden Sie im Verzeichnis `...\Examples\Tutorial`.

Im Verzeichnis `Examples` finden Sie viele XML-Dateien, mit denen Sie experimentieren können. Im Verzeichnis `Tutorial` sind alle Dateien abgelegt, die in dieser Anleitung verwendet werden.

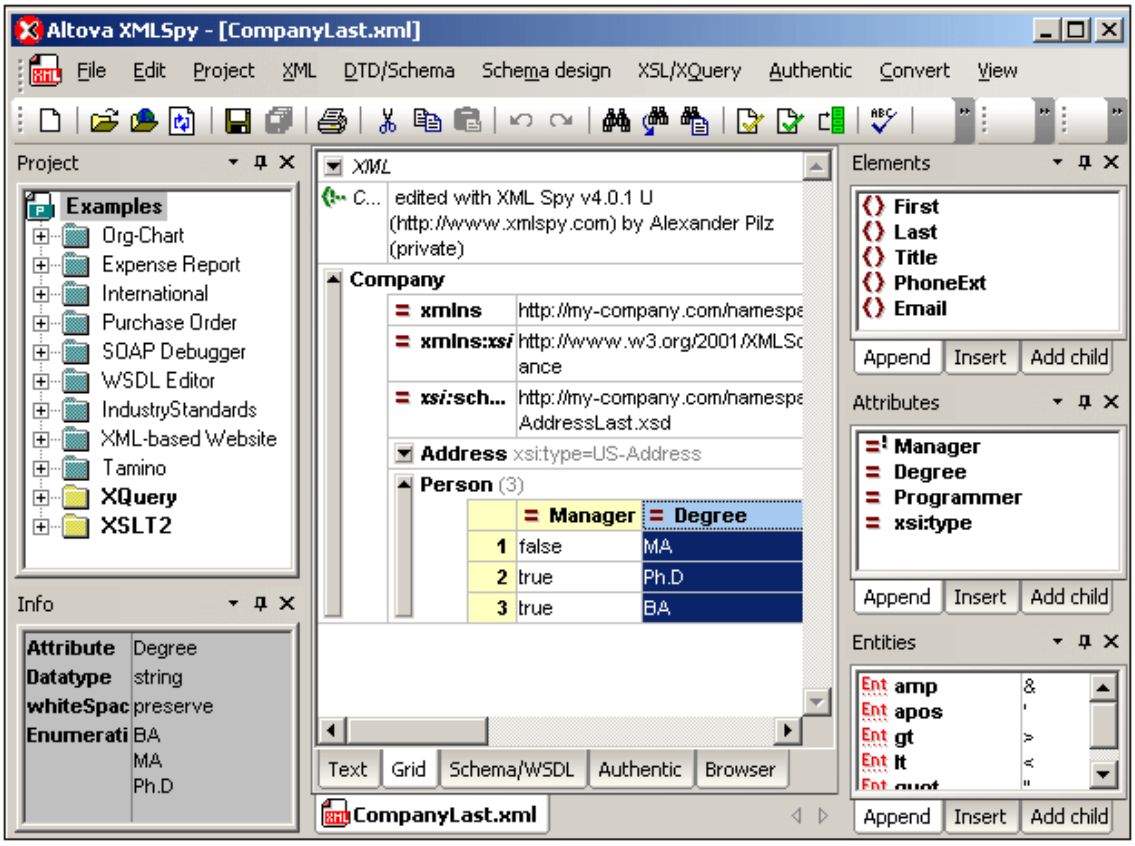
Das Verzeichnis `Template` **(Vorlagen)** enthält sämtliche XML-Vorlagen, die verwendet werden, wenn Sie die Menüoption **Datei | Neu** aufrufen. Diese Dateien enthalten alle nötigen Daten (Namespace und XML-Deklarationen) die Sie benötigen, um mit dem XML-Dokument arbeiten zu können.

2.1.1 Die XMLSpy-Benutzeroberfläche

Die Oberfläche von XMLSpy 2005 ist vertikal in drei Bereiche gegliedert. Im mittleren Bereich stehen verschiedene Ansichten für Ihr Dokument bereit. Die beiden seitlichen Bereiche enthalten Informationen, Editierhilfen und Funktionen zur Dateiverwaltung.

- Das **Projekt-** und das **Info-Fenster** befinden sich auf der linken Seite.
- Der mittlere Teil besteht aus dem **Hauptfenster**. Hier bearbeiten Sie Ihr XML-Dokument. Sie können zwischen verschiedenen Ansichten wählen: Textansicht, Grid-Ansicht, Schema/WSDL Design-Ansicht, Authentic-Ansicht und Browser-Ansicht. Diese Ansichten werden in den entsprechenden Abschnitten dazu näher beschrieben.
- Auf der rechten Seite des Bildschirmes befinden sich drei **Eingabenhilfen**-Fenster. Hier können Sie Elemente, Attribute oder Entities einfügen oder anhängen. Die verfügbaren

Einträge ändern sich je nachdem, welche Auswahl in der Grid-Ansicht vorgenommen wurde, bzw. je nach Positionierung des Cursors in der XML-Datei.



Die Einzelheiten der Benutzeroberfläche werden im Laufe des Handbuchs näher beschrieben. Beachten Sie, dass die Oberfläche sich dynamisch je nach gewählter Ansicht ändert und je nachdem, welches Dokument im Hauptfenster aktiv ist.

2.1.2 Erstellen eines einfachen XML-Schemas

Ein XML-Schema beschreibt die Struktur eines XML-Dokuments. Ein XML-Dokument kann gegen ein XML-Schema validiert werden, um zu überprüfen, ob es mit den im Schema definierten Vorgaben übereinstimmt und somit gültig ist. Falls dies nicht der Fall ist, ist es ungültig. Mit Hilfe von Schemas kann der Verfasser eines Dokuments die Struktur und den zulässigen Inhalt eines XML-Dokuments definieren und überprüfen, ob das XML-Dokument gültig ist.

Ein XML-Schema hat eine komplexe Struktur und Syntax. Da es sich bei einem XML-Schema selbst um ein XML-Dokument handelt, muss es den Regeln der XML-Schema-Spezifikation entsprechen. In XMLSpy 2005 können Sie in der Schema/WSDL Design-Ansicht mit Hilfe von Drag & Drop-Techniken gültige XML-Schemas auf einfache Art erstellen. Das von Ihnen erzeugte XML-Schema-Dokument kann auch in der Text- und der Grid-Ansicht bearbeitet werden, viel einfacher lässt sich dies jedoch in der Schema/WSDL-Ansicht bewerkstelligen.

In diesem Abschnitt lernen Sie, wie man ein XML-Schemas in der Schema/WSDL Design-Ansicht bearbeitet. Insbesondere wird auf folgende Punkte eingegangen:

© 2005 Altova GmbH

- Erstellen einer neuen Schema-Datei
- Definieren von Namespaces für das Schema
- Definieren eines einfachen Content Model
- Hinzufügen von Elementen zum Content Model über Kontextmenüs und mittels Drag & Drop
- Konfiguration der Content Model-Ansicht

Nachdem Sie mit der Erstellung des Schema-Grundgerüsts fertig sind, können Sie zum nächsten Abschnitt des Tutorials übergehen, in dem Sie erfahren, wie man mit komplexeren XML-Schema-Funktionen in XMLSpy 2005 arbeitet. Auf diesen Abschnitt folgt ein Abschnitt über Schema-Navigation und Dokumentation in XMLSpy 2005.

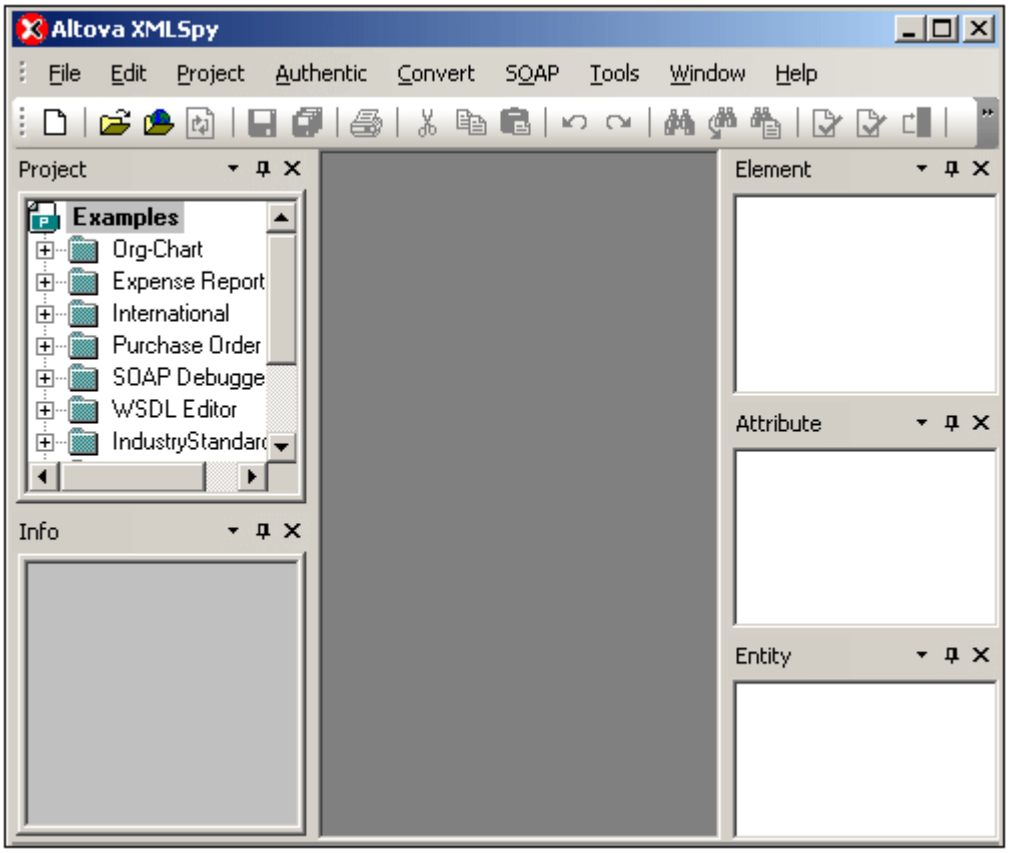
In diesem Abschnitt verwendete Befehle
In diesem Abschnitt arbeiten Sie ausschließlich in der Schema/WSDL-Ansicht. Dabei werden die folgenden Befehle verwendet:

Diagramm anzeigen (oder Content Model-Ansicht anzeigen). Dieses Symbol befindet sich links von allen globalen Komponenten in der Schema-Übersicht. Wenn Sie auf das Symbol klicken, wird das Content Model der dazugehörigen globalen Komponente angezeigt.

Anlegen einer neuen XML-Schema-Datei

Um in XMLSpy 2005 eine neue XML-Schema-Datei anzulegen, müssen Sie zuerst XMLSpy 2005 starten und dann ein neues XML-Schema (`.xsd`)-Dokument erstellen.

Starten von XMLSpy 2005
Doppelklicken Sie zum Starten von XMLSpy 2005 auf das XMLSpy 2005-Symbol auf Ihrem Desktop oder rufen Sie XMLSpy 2005 über **Start | Programme** auf. XMLSpy 2005 wird mit einem leeren Hauptfenster gestartet, in dem kein Dokument geöffnet ist.

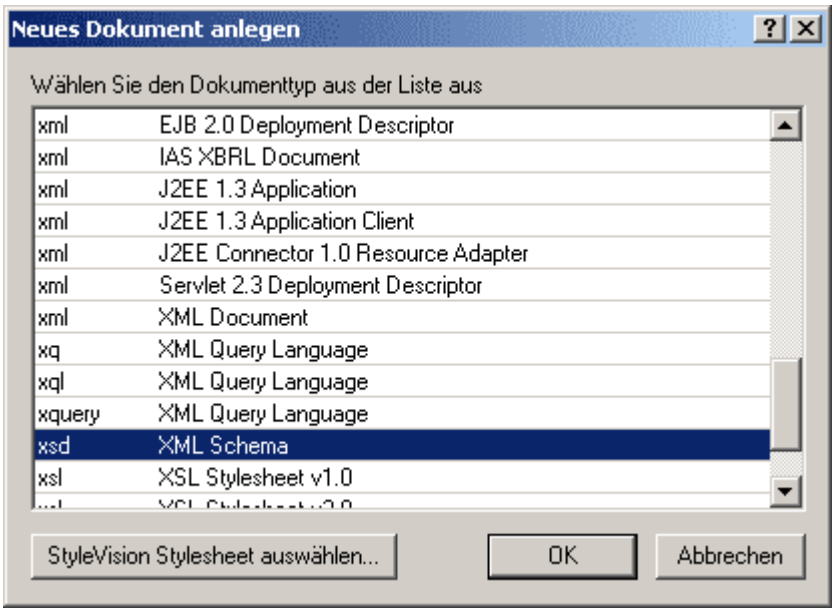


Beachten Sie, dass sich die Benutzeroberfläche in drei Teile gliedert: (i) das Projekt- und das Info-Fenster auf der linke Seite; (ii) das Hauptfenster in der Mitte; und (iii) die Eingabehilfen auf der rechten Seite.

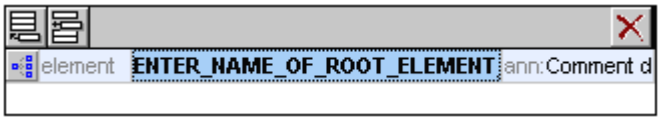
© 2005 Altova GmbH

So legen Sie eine neue Schema-Datei an:

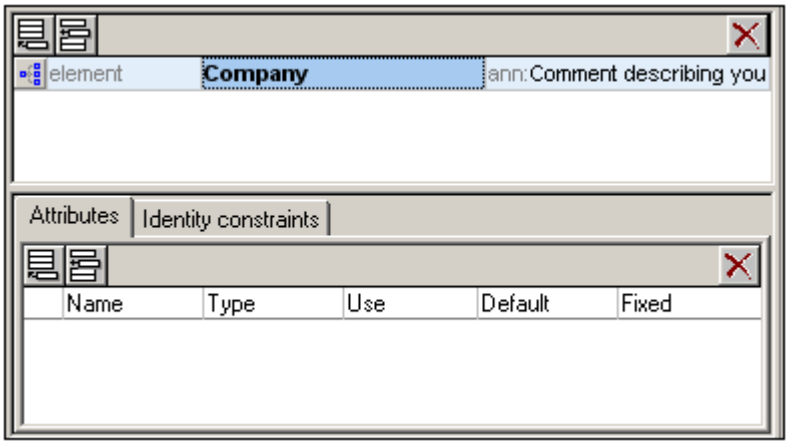
1. Wählen Sie die Menüoption **Datei | Neu**. Das Dialogfeld "Neues Dokument anlegen" wird angezeigt.



2. Wählen Sie im Dialogfeld den Eintrag `.xsd W3C Schema`**.** Bestätigen Sie mit OK. Eine leere Schema Datei wird im Hauptfenster in der Schema/WSDL Design-Ansicht angezeigt. Die Schema/WSDL Design-Ansicht wiederum hat zwei Ansichten: die **Schema-Übersicht**, die eine Übersicht aller globalen Komponenten enthält, und die **Content Model-Ansicht**, die eine Ansicht des Content Model der einzelnen globalen Komponenten enthält. Beim Erstellen einer neuen XML-Schema-Datei wird die Schema/WSDL-Ansicht in der Schema-Übersicht geöffnet. Sie werden aufgefordert, den Namen des Root-Elements anzugeben.



3. Klicken Sie in das markierte Feld, und geben Sie `Company` ein. Bestätigen Sie mit der Eingabetaste. `Company` ist somit das "Root" Element dieses Schemas und gleichzeitig ein globales Element. Die im Hauptfenster angezeigte Ansicht (*siehe Abbildung unten*) nennt sich Schema-Übersicht und enthält eine Liste aller globalen Komponenten im oberen Bereich Fenster; die Attribute und Identity Constraints der ausgewählten Komponente werden im unteren Fenster angezeigt. (Sie können das Content Model einzelner globaler Komponenten ansehen und bearbeiten, indem Sie auf das Symbol "Diagramm anzeigen" klicken, das links von der globalen Komponente zu sehen ist.)



4. Geben Sie im Annotationsfeld des Elements `Company` die Beschreibung für das Element ein, in diesem Fall `Root element`.
5. Klicken Sie auf die Menüoption **Datei | Speichern** und benennen Sie Ihr Schema (z.B. `AddressFirst.xsd`).

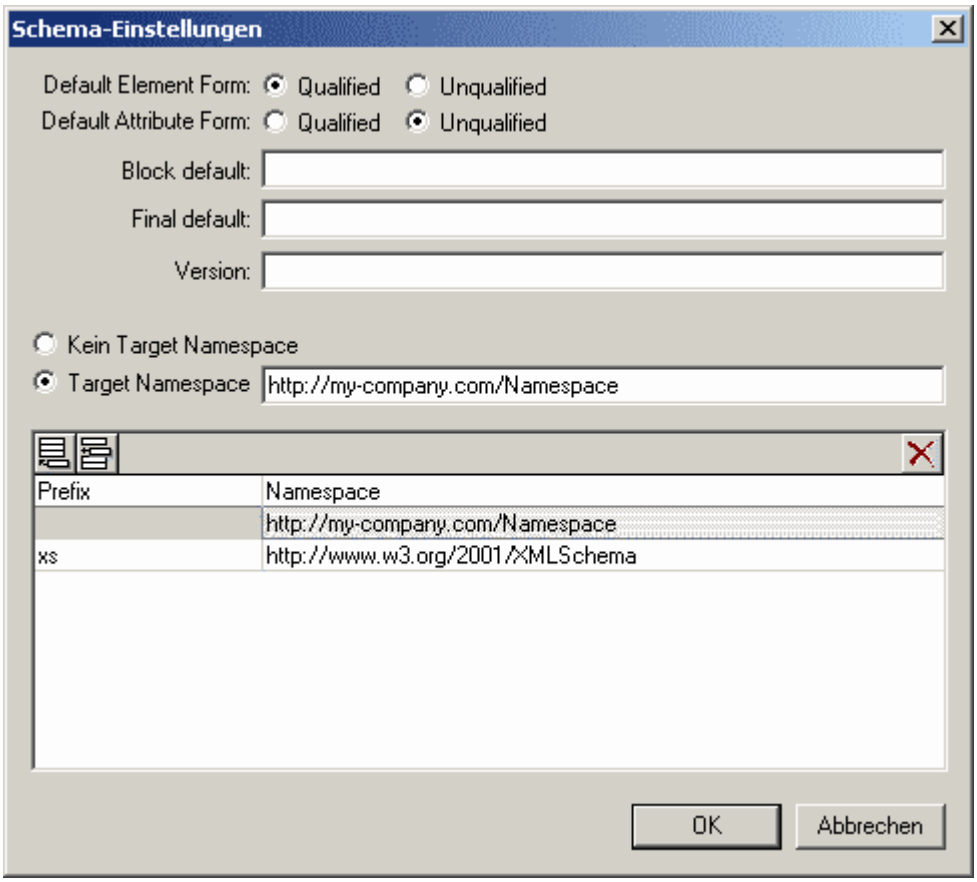
Definieren von Namespaces

XML Namespaces spielen eine wichtige Rolle in XML-Schemas und XML-Dokumenten. Ein XML-Schema muss den XML-Schema-Namespace referenzieren und kann optional auch einen Target Namespace für das XML-Instanzdokument definieren. Als Schema-Autor müssen Sie festlegen, wie Sie diese beiden Namespaces definieren (im Grunde mit Hilfe von Präfixen).

Sie werden nun im XML-Schema, das Sie erstellen, einen Target Namespace für das XML-Instanzdokument definieren. (Die erforderliche Referenz zum XML Schema Namespace wird von XMLSpy 2005 automatisch erstellt, wenn Sie ein neues XML Schema-Dokument anlegen.) So erstellen Sie einen Target Namespace:

1. Wählen Sie die Menüoption **Schema-Design | Schema-Einstellungen**, um das Dialogfeld "Schema-Einstellungen" aufzurufen.

© 2005 Altova GmbH



2. Klicken Sie auf das Target Namespace-Optionsfeld und geben Sie `http://my-company.com/Namespace` ein. In XMLSpy 2005 wird der von Ihnen als Target Namespace eingegebene Namespace als Default Namespace des XML-Schema-Dokuments erstellt und in der Liste der Namespaces im unteren Bereich des Dialogfelds angezeigt.
3. Bestätigen Sie mit OK.

Hinweis:

- Der XML Schema Namespace wird von XMLSpy 2005 automatisch erstellt und erhält das Präfix `xs:`.
- Wenn Sie das XML-Instanzdokument erstellen, muss sein Dokumentmodell den im XML-Schema definierten Target Namespace haben, um gültig zu sein.

Definieren eines Content Model

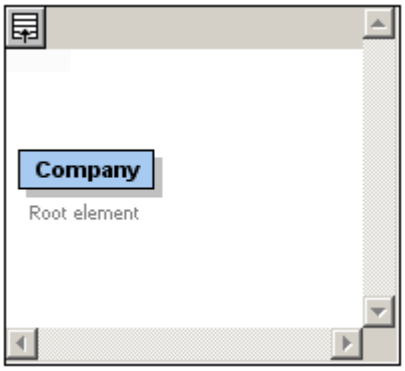
Sie haben in der Schema-Übersicht bereits ein globales Element mit dem Namen `Company` erstellt. Dieses Element soll ein `Address`-Element und eine unbegrenzte Anzahl an `Person`-Elementen enthalten, nämlich das Content Model. Globale Elemente, die Content Models haben können, sind Elemente, complexTypes und Elementgruppen.

In XMLSpy 2005 wird das Content Model einer globalen Komponente in der Content Model-Ansicht der Schema/WSDL-Ansicht angezeigt. Um das Content Model einer globalen Komponente anzeigen und editieren zu können, klicken Sie auf das Symbol "Diagramm anzeigen" links neben der globalen Komponente.

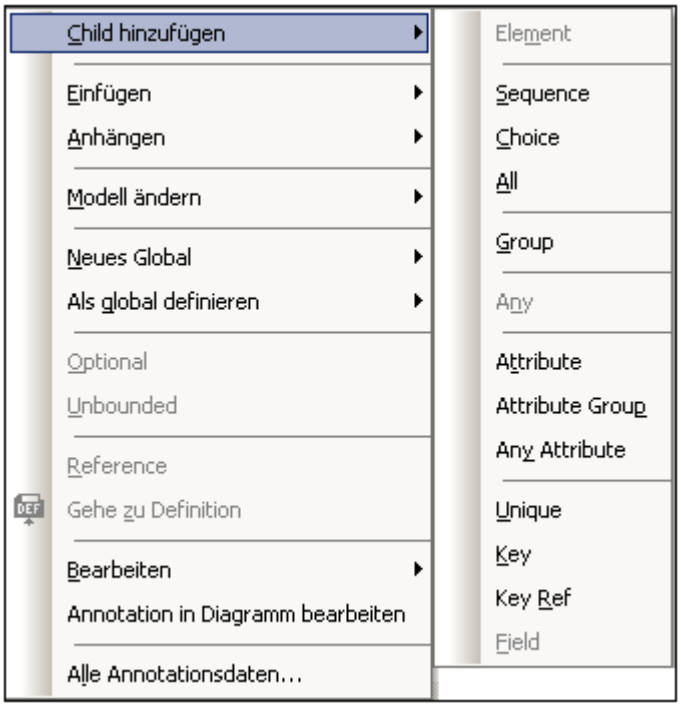
In diesem Abschnitt werden Sie das Content Model des Elements `Company` erstellen.

Erstellen eines einfachen Content Model
So erstellen Sie das Content Model des Elements `Company`:

1. Klicken Sie in der Schema-Übersicht auf das Symbol "Diagramm anzeigen" links neben dem Element `Company`. Hierdurch wird das Content Model des Elements `Company` angezeigt, das derzeit noch leer ist. (Alternativ dazu können Sie auch auf den Eintrag `Company` im Komponentenfenster (oberste Eingabehilfe) doppelklicken, um das Content Model anzuzeigen.)

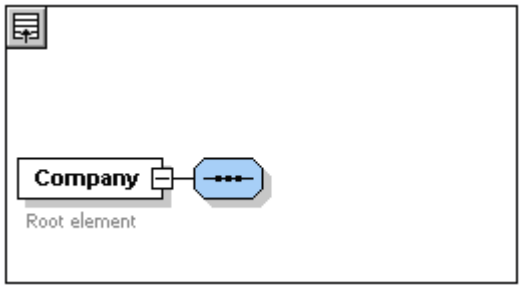


2. Ein Content Model besteht aus Kompositoren und Komponenten. Die Kompositoren definieren die Beziehung zwischen zwei Komponenten. Sie müssen nun im `Company` Content Model einen Child-Kompositor zum Element `Company` hinzufügen, um ein Child-Element hinzufügen zu können. Rechtsklicken Sie dazu auf das Element "Company" und wählen Sie im Kontextmenü die Option **Child hinzufügen | Sequence**. (Sequence, Choice und All sind die drei Kompositoren, die in einem Content Model verwendet werden können.)

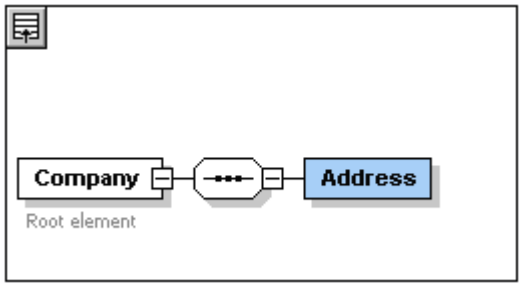


Daraufhin wird der Sequence-Kompositor eingefügt, der definiert, dass die Komponenten, die darauf folgen in der angegebenen Reihenfolge erscheinen müssen.

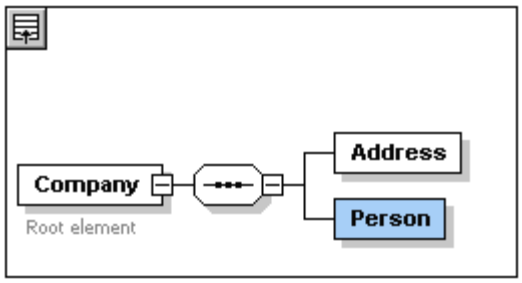
 © 2005 Altova GmbH



3. Rechtsklicken Sie auf den Sequence-Kompositor und wählen Sie **Child hinzufügen | Element**. Daraufhin wird ein unbenanntes Element hinzugefügt.
4. Benennen Sie das Feld `Address` und bestätigen Sie mit der Eingabetaste.

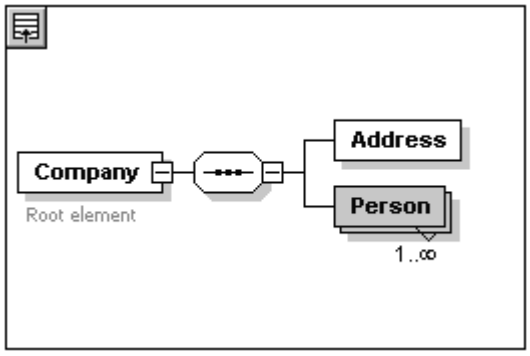


5. Rechtsklicken Sie erneut auf den Sequence-Kompositor und wählen Sie **Child hinzufügen | Element**, benennen Sie die neue Elementkomponente `Person`.



Sie haben nun ein Schema mit einem Eintrag für die Firmenadresse und einem Eintrag für einen Angestellten (`Person`) definiert. Wir müssen nun die Anzahl der `Person`-Elemente erhöhen.

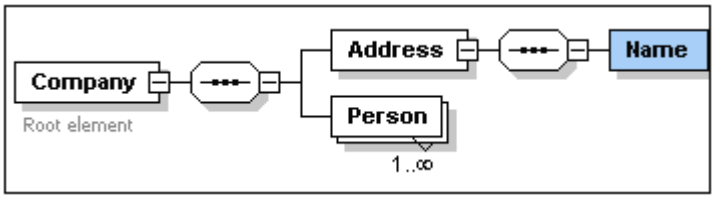
6. Klicken Sie mit der rechten Maustaste auf das Element `Person` und wählen Sie **Unbounded** aus dem Kontextmenü.
Unter dem Element `Person` wird nun die Anzahl der zulässigen Instanzen angezeigt: 1 bis unendlich.



Sie können alternativ dazu in der Details-Eingabehilfe die Felder `minOcc` und `maxOcc` direkt editieren und die Anzahl der zulässigen Instanzen angeben, in diesem Fall 1 bis unendlich.

Hinzufügen von zusätzlichen Ebenen zur Struktur des Content Model
Bisher enthält das Content Model, das Sie erstellt haben, eine Ebene: eine Child-Ebene für das Element `company`, das die Elemente `Address` und `Person` enthält. Nun wollen wir den Inhalt des Elements `Address` definieren, und zwar die Subelemente `Name`, `Street` und `City`. Dies ist eine zweite Ebene. Wir müssen also zum Element `Address` einen Child-Kompositor hinzufügen und anschließend die Elementkomponenten selbst. Gehen Sie dabei folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf das Element `Address`, um das Kontextmenü zu öffnen, und wählen Sie **Child hinzufügen | Sequence**.
2. Klicken Sie mit der rechten Maustaste auf den **Sequence-**Kompositor und wählen Sie **Child hinzufügen | Element**. Benennen Sie die neu erstellte Elementkomponente mit "Name".



ComplexTypes, simpleTypes und XML Schema-Datentypen
Bisher haben wir noch keinen Elementtyp explizit definiert. Wenn Sie sich die Textansicht Ihres Schemas jetzt ansehen (*siehe unten)*, werden Sie feststellen, dass jedes Mal, wenn ein Sequence-Kompositor eingefügt wurde, das Element `xs:sequence` im Element `xs:complexType` eingefügt wurde. Kurz gesagt, die Elemente `Company` und `Address` sind ComplexTypes, da sie Child-Elemente enthalten. Ein ComplexType ist ein Element, das Attribute oder Elemente enthält.

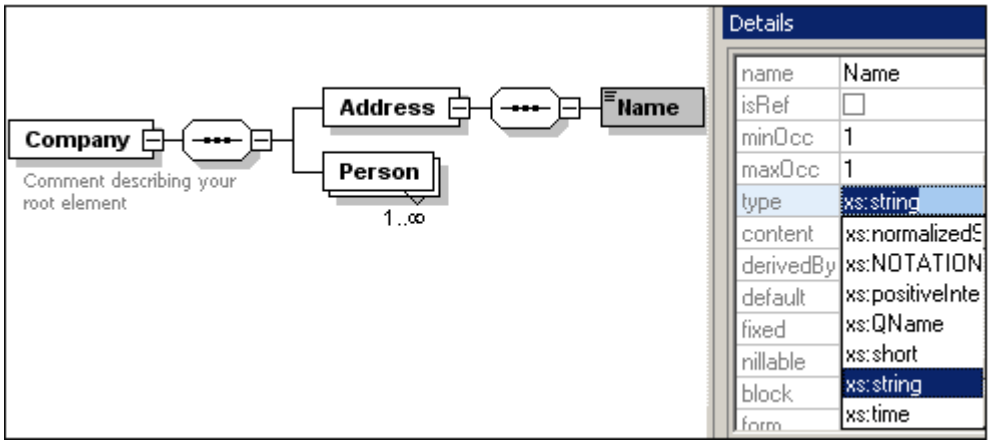
```
<xs:element name="Company">
  <xs:annotation>
    <xs:documentation>Root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Address">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Person"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

SimpleType-Elemente dagegen enthalten nur Text und haben keine Attribute. Beim Text kann es sich um Strings, Daten, Zahlen usw. handeln. Wir möchten, dass das Child-Element `Name` von `Address` nur Text enthält. Es ist ein SimpleType, dessen Textinhalt wir durch Restriktion auf einen String einschränken möchten. Dies lässt sich durch Verwendung des XML-Schema-Datentyps `xs:string` bewerkstelligen. Um das Element `Name` als diesen Datentyp zu definieren, gehen Sie folgendermaßen vor.

1. Klicken Sie auf `Name`, um das Element auszuwählen.

 © 2005 Altova GmbH

2. Klicken Sie in der Details-Eingabehilfe auf das Listenfeld `type` in der mittleren Eingabehilfe und wählen Sie `xs:string` aus der Auswahlliste.



Links oben im Element `Name` erscheint ein Symbol

, das anzeigt, dass dieses Element Textdaten enthält. Beachten Sie, dass sowohl `minOcc` als auch `maxOcc` den Wert 1 hat, d.h. dieses Element hat nur eine Instanz.

In der Textdarstellung sieht das Element `Name` folgendermaßen aus:

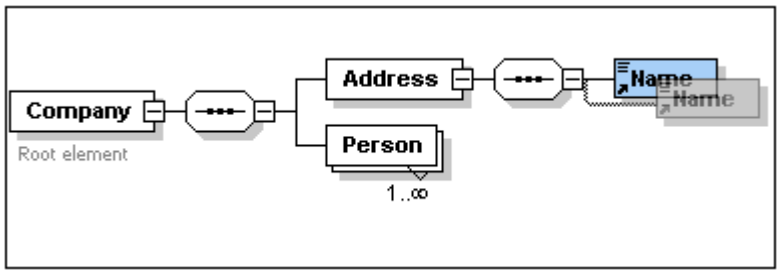
```
<xs:element name="Name" type="xs:string"/>
```

Hinweis: Ein einfaches Type-Element kann jedem beliebigen von mehreren XML-Schema-Datentypen angehören. In all diesen Fällen erscheint das Symbol für Textinhalt nur im Elementfeld.

Hinzufügen von Elementen mit Drag & Drop

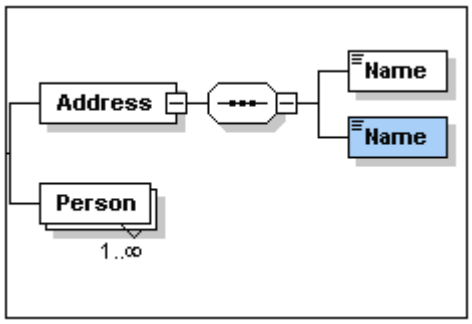
Sie haben Elemente bisher über das Kontextmenü hinzugefügt, das erscheint, wenn Sie mit der rechten Maustaste auf ein Element oder einen Kompositor klicken. Schneller lassen sich Elemente aber mit Drag & Drop erstellen. In diesem Abschnitt lernen Sie, wie man das Element `Address` mit der Drag & Drop-Methode erstellt. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie auf das Element `Name`, halten Sie die Maustaste sowie die **STRG**-Taste gedrückt und ziehen Sie die Maus leicht nach unten.
   Die Markierung mit einem +Zeichen erscheint. Hierdurch wird angezeigt, dass das Element kopiert werden kann. Es erscheint eine Kopie des Elements sowie eine Konnektorlinie, sodass Sie sehen, wo das Element erstellt wird.

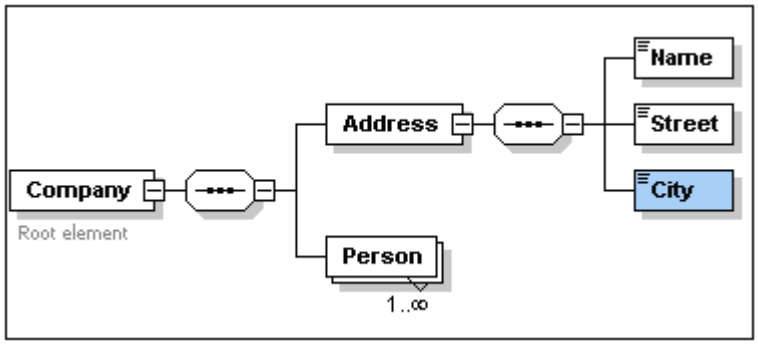


2. Lassen Sie nun die Maustaste und die **STRG**-Taste los um das neue Element anzulegen. Wurde das Element an der falschen Stelle angelegt, so ziehen Sie es einfach mit der Maus unter das Element `Name` und lassen Sie die Maustaste los. Damit haben Sie nun eine Kopie des Originalelements erstellt, in diesem Fall des Elements

`Name`.



3. Doppelklicken Sie in das Elementfeld und geben Sie `Street` ein, um den Namen des Elements zu ändern.
4. Wiederholen Sie obige Schritte um ein weiteres Element zu erzeugen. Benennen Sie dieses Feld um und geben Sie ihm den Namen `City`.
   Das Content Model sollte nun folgenden Inhalt haben:



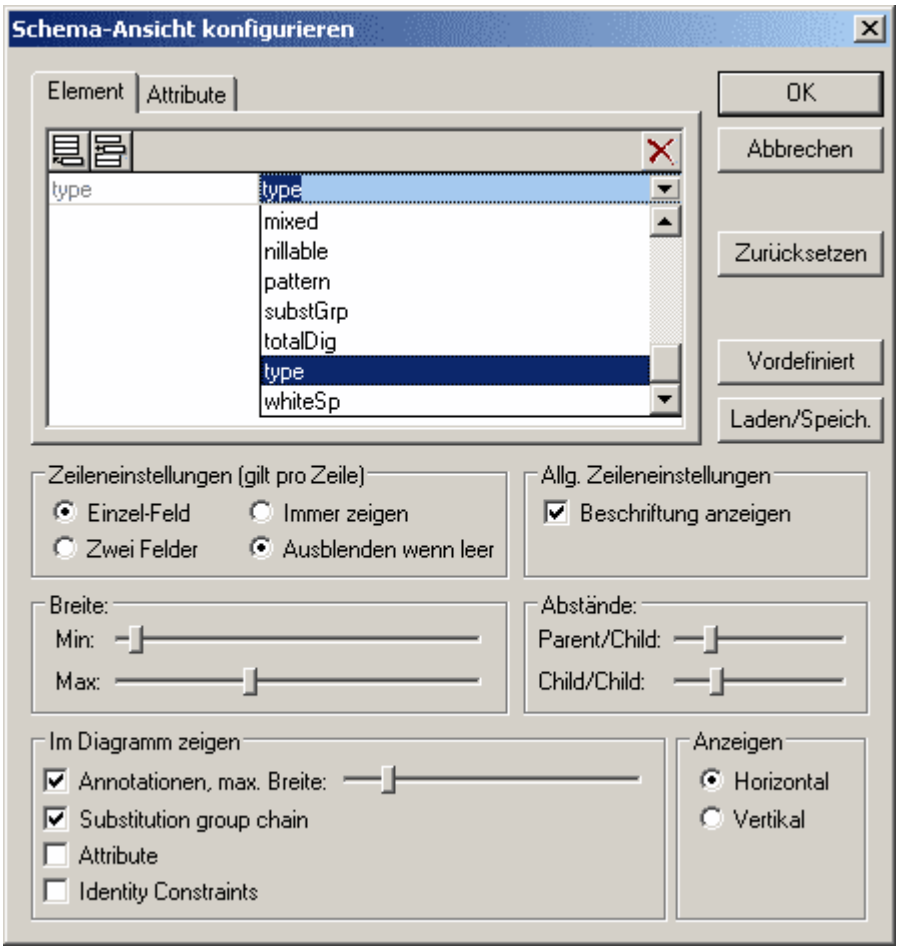
Das Element `Address` enthält eine Sequence mit einem Element `Name`, einem Element `Street` und einem Element `City`, und zwar in der obigen Reihenfolge.

Konfigurieren der Content Model-Ansicht

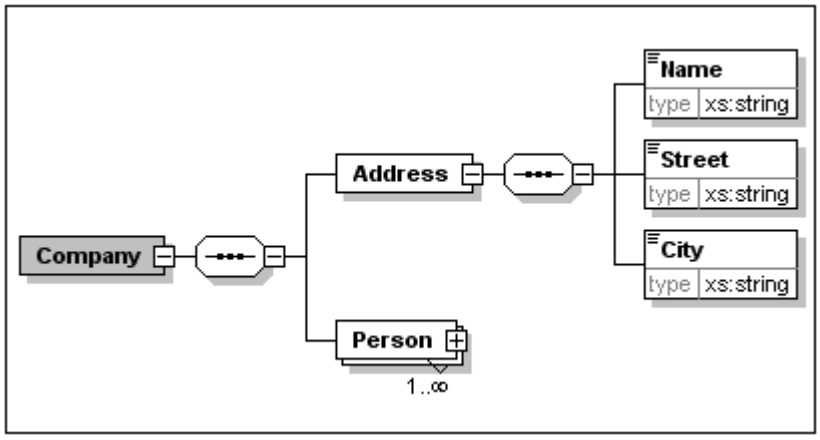
Sie können jetzt das Aussehen der Content Model-Ansicht konfigurieren.
Gehen Sie dazu folgendermaßen vor:

1. Gehen Sie zur Content Model-Ansicht einer Komponente, um den Befehl "Ansicht konfigurieren" zu aktivieren.
2. Wählen Sie die Menüoption **Schema-Design | Ansicht konfigurieren**, um das Dialogfeld "Schema-Ansicht konfigurieren" aufzurufen.

© 2005 Altova GmbH

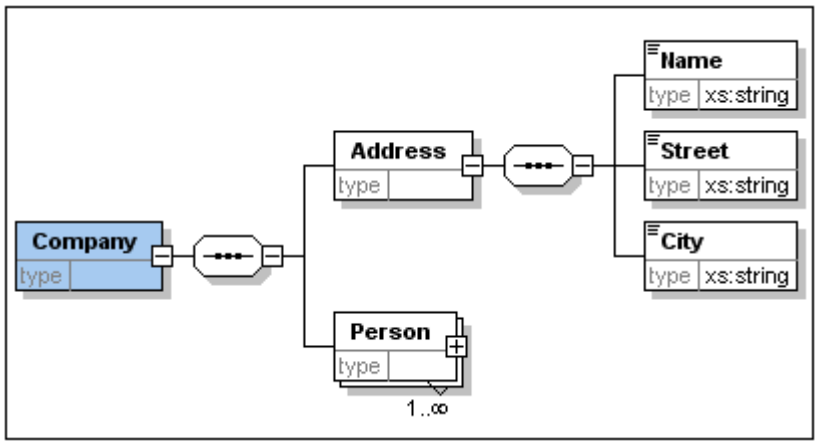


3. Klicken Sie im Element/Attribut-Fenster auf dem Register "Element" auf das Symbol **Anhängen** um eine Zeile zur Beschreibung einer Eigenschaft zu jedem Elementfeld hinzuzufügen.
4. Wählen Sie `type` aus der Auswahlliste (oder doppelklicken Sie in die Zeile und tippen Sie "`type`"). Dadurch wird der Datentyp jedes Elements in der Content Model-Ansicht angezeigt.
5. Wählen Sie im Bereich "Zeileneinstellungen (Gilt pro Zeile)" die Option "Ausblenden wenn leer", damit die Beschreibung des Datentyps im Elementfeld ausgeblendet wird, wenn das Element keinen Datentyp hat (z.B. wenn das Element ein ComplexType ist).



Beachten Sie, dass die Beschreibungszeile für den Typ bei den Elementen `Name`, `Street` und `City` aufscheint, da es sich dabei um SimpleTypes vom Typ `xs:string` handelt, nicht aber bei den ComplexType-Elementen, da die Option "Ausblenden wenn leer" aktiv ist.

6. Aktivieren Sie im Bereich "Zeileneinstellungen (Gilt pro Zeile)" die Option "Immer zeigen".
7. Klicken Sie auf **OK**, um die Änderungen zu speichern.



Beachten Sie, dass die Beschreibungszeile für den Datentyp immer angezeigt wird - selbst in Elementfeldern von ComplexTypes, in denen keine Werte in dieser Zeile angezeigt werden.

Hinweis:

- Die Eigenschaftsbeschreibungszeilen können editiert werden. Werte, die Sie dort eingeben, werden Teil der Elementdefinition.
- Die Einstellungen, die Sie im Dialogfeld "Schema-Ansicht konfigurieren" vornehmen, gelten für die Ausgabe der Schema-Dokumentation ebenso wie für die Druckausgabe.

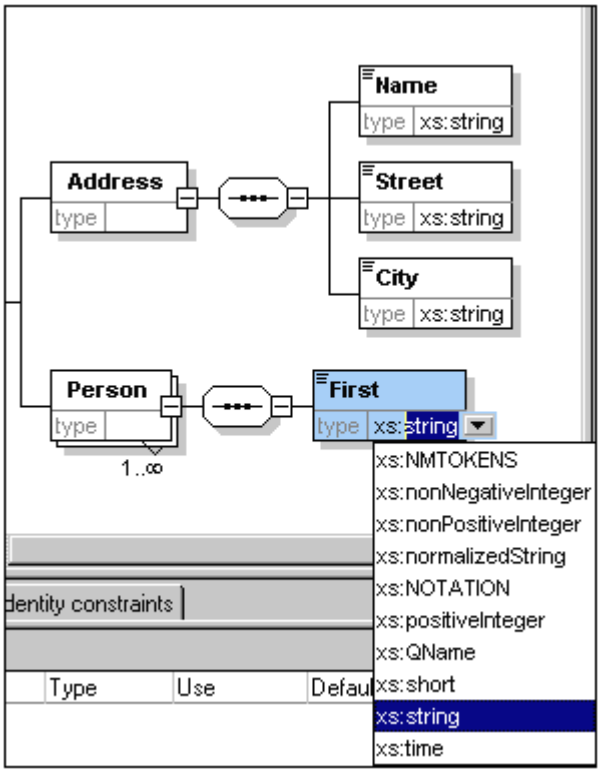
Fertigstellen des Schemas

Nachdem Sie nun den Inhalt des Elements `Address` definiert haben, müssen Sie den Inhalt des Elements `Person` definieren. Das Element `Person` soll die folgenden Child-Elemente - alle vom Typ SimpleType - enthalten: `First`, `Last`, `Title`, `PhoneExt` und `Email`. Alle diese Elemente sind obligatorisch mit Ausnahme des Elements `Title` (das optional ist), und alle Elemente müssen in der oben angegebenen Reihenfolge aufscheinen. Alle Elemente sollen vom Datentyp `xs:string` sein, mit Ausnahme von `PhoneExt`, das vom Datentyp `xs:integer` sein muss und maximal 2 Stellen haben darf.

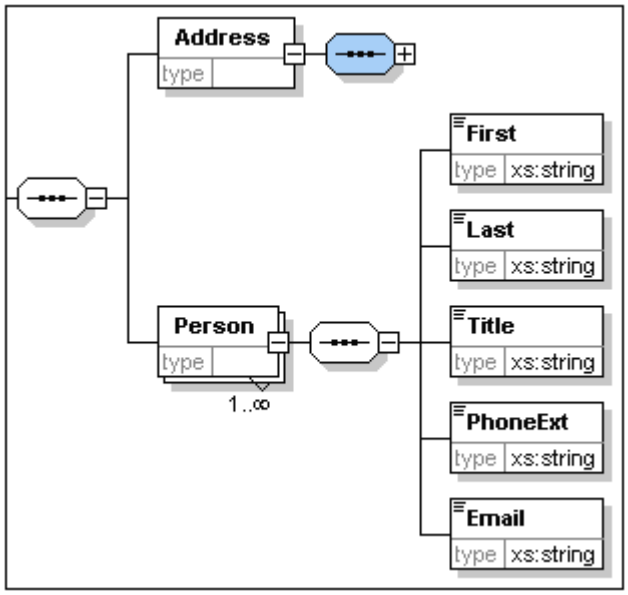
Um das Content Model für `Person` zu erstellen, gehen Sie folgendermaßen vor:

1. Rechtsklicken Sie auf das Element `Person`, um das Kontextmenü aufzurufen und wählen Sie **Child hinzufügen | Sequence**. Der Sequence-Kompositor wird eingefügt.
2. Klicken Sie mit der rechten Maustaste auf den Sequence-Kompositor und wählen Sie **Child hinzufügen | Element**.
3. Nennen Sie das Element `First` und betätigen Sie die **Tabulatortaste** Ihrer Tastatur. Sie gelangen automatisch in das Feld `type`.

© 2005 Altova GmbH



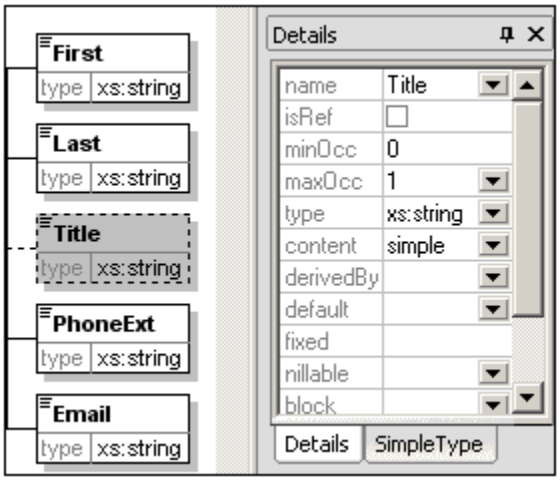
4. Wählen Sie den Eintrag `xs:string` aus der Auswahlliste oder geben Sie ihn in das Wertefeld von `type` ein.
5. Erzeugen Sie weitere vier Elemente mittels der Drag & Drop Methode. Benennen Sie diese der Reihe nach `Last`, `Title`, `PhoneExt` und `Email`.



Hinweis: Sie können mehrere Elemente auswählen, indem Sie die **Strg**-Taste gedrückt halten, während Sie die Elemente anklicken.

Ein Element optional machen

Rechtsklicken Sie auf das Element `Title` und wählen Sie **Optional** im Kontextmenü. Die Umrahmung des Elementes wechselt von durchgezogener Linie auf gestrichelt. Hierdurch wird angezeigt, dass dieses Element optional ist.

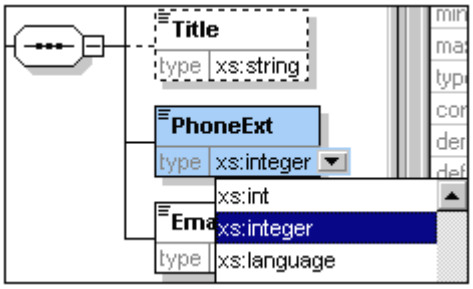


Sie sehen in der Details-Eingabehilfe, dass `minOcc=0` und `maxOcc=1` ist, d.h. das Element ist optional. Alternativ dazu können Sie das Element auch über das Kontextmenü optional machen und für `minOcc=0` einstellen, um das Element zu einem optionalen zu machen.

Den Inhalt eines Elements einschränken

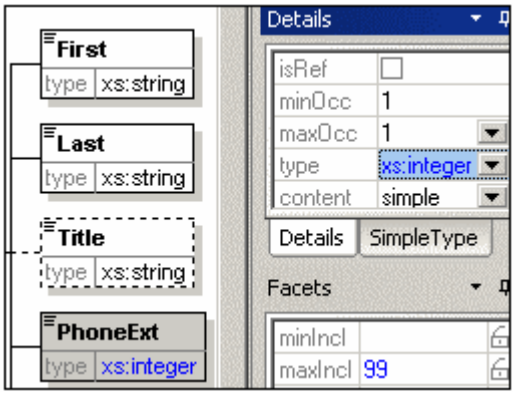
Um das Element `PhoneExt` als Typ `xs:integer` mit maximal 2 Stellen zu definieren, gehen Sie folgendermaßen vor:

1. Doppelklicken Sie in das Feld `type` unter dem Element `PhoneExt` und wählen Sie den Eintrag `xs:integer` aus der Auswahlliste oder geben Sie ihn ein.



   Änderungen werden in der Facets-Eingabehilfe angezeigt.
2. Doppelklicken Sie in der Facets-Eingabehilfe in das Feld `maxIncl`, geben Sie `99` ein und bestätigen Sie dies durch Drücken der Eingabetaste.

© 2005 Altova GmbH



Hierdurch wird definiert, dass alle Durchwahlnummern bis inkl. 99 gültig sind.

3. Wählen Sie die Menüoption **Datei | Speichern** zum Speichern der Änderungen im Schema.

Hinweis:

- Beim Auswählen eines XML-Schema-Datentyps vom Typ simpleType (z.B. `xs:string` und `xs:date`) wird das Content Model automatisch in der Details-Eingabehilfe auf simple (`content = simple`) gesetzt.
- Beim Hinzufügen eines Kompositors zu einem Element (`selection`, `choice` oder `all`), wird das Inhaltsmodell automatisch auf complex (`content = complex`) gesetzt.
- Das verwendete Schema ist als `AddressFirst` im Verzeichnis `Examples\Tutorial` Ihres XMLSpy 2005-Anwendungsordners abgelegt.

2.1.3 Komplexere XML-Schema-Definitionen

Nachdem Sie nun ein einfaches Schema erstellt haben, können wir uns den komplizierteren Aspekten der Schema-Entwicklung widmen. In diesem Abschnitt werden folgende Punkte behandelt:

- Arbeiten mit complexTypes und simpleTypes, die später als die Typen von Schema-Elementen verwendet werden können.
- Erstellen von globalen Elementen und Referenzieren dieser Elemente von anderen Elementen aus
- Erstellen von Attributen und deren Eigenschaften, einschließlich Enumerationswerten.

Wir beginnen diesen Abschnitt mit dem einfachen Schema **`AddressFirst.xsd`**, das Sie im ersten Teil dieses Tutorials erstellt haben.

In diesem Abschnitt verwendete Befehle
In diesem Abschnitt des Tutorials wird ausschließlich die Schema/WSDL-Ansicht verwendet. Dabei kommen die folgenden Befehle zum Einsatz:

Diagramm anzeigen (oder Content Model-Ansicht anzeigen). Mit diesem Symbol, das in der Schema-Übersicht links von allen globalen Komponenten angezeigt wird, sehen Sie das Content Model der dazugehörigen globalen Komponente

Globale Elemente anzeigen. Dieses Symbol befindet sich in der linken oberen Ecke des Content Model. Mit diesem Befehl kommen Sie zurück zur Schema-Übersicht, in der alle globalen Komponenten angezeigt werden.

Anhängen. Mit diesem Symbol (in der linken oberen Ecke der Schema-Übersicht) können Sie eine globale Komponente hinzufügen.

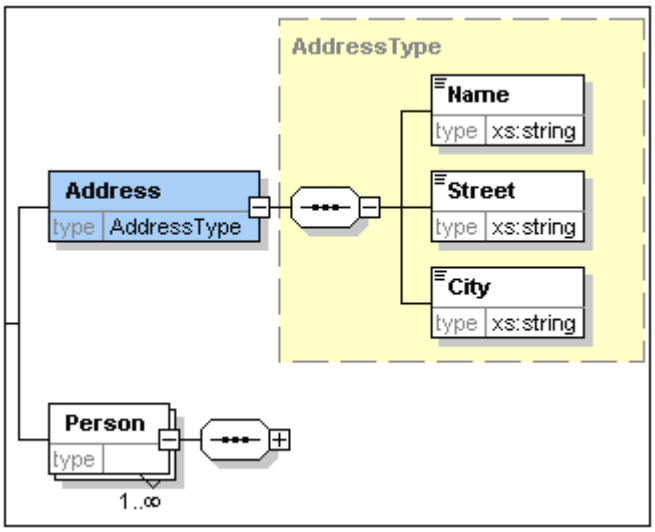
Arbeiten mit ComplexTypes und SimpleTypes

Nachdem Sie ein Element definiert haben, möchten Sie es möglicherweise auch an einer anderen Stelle im Schema verwenden. Dies erreichen Sie dadurch, dass Sie diese Elementdefinition als globalen ComplexType oder als globales Element erstellen. In diesem Abschnitt werden Sie mit globalen ComplexTypes arbeiten. Wir werden zuerst einen ComplexType auf globaler Ebene erstellen und diesen dann erweitern, um ihn in einem Content Model verwenden zu können. Mehr über globale Elemente erfahren Sie weiter hinten in diesem Tutorial.

Erstellen einer globalen Komponente

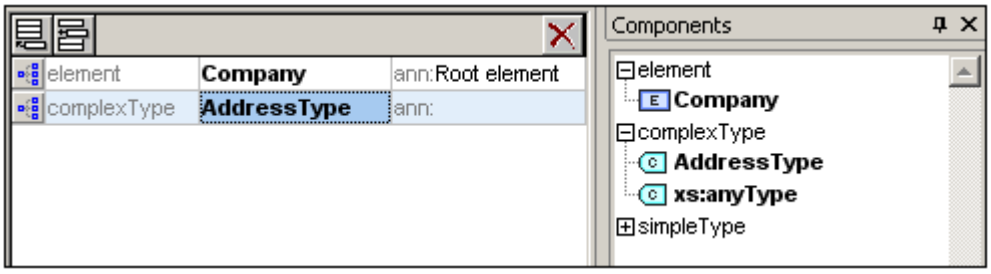
Das Basiselement `Address`, das wir zuvor definiert haben (und das die Elemente `Name`, `Street` und `City` enthält), kann in verschiedenen Adressformaten wiederverwendet werden. Wir wollen daher diese Elementdefinition als wiederverwendbaren ComplexType erstellen. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie in der Content Model-Ansicht mit der rechten Maustaste auf das Element `Address`.
2. Wählen Sie im Kontextmenü nun die Option **Als global definieren | Complex Type**. Es wird ein globaler ComplexType mit dem Namen `AddressType` erstellt, und dem Element `Address` im Content Model `Company` wird dieser Typ zugewiesen. Der Inhalt des Elements `Address` ist das Content Model von `AddressType`, das in einem gelb hinterlegten Kasten angezeigt wird. Beachten Sie, dass der Datentyp des Elements `Address` nun `AddressType` ist.

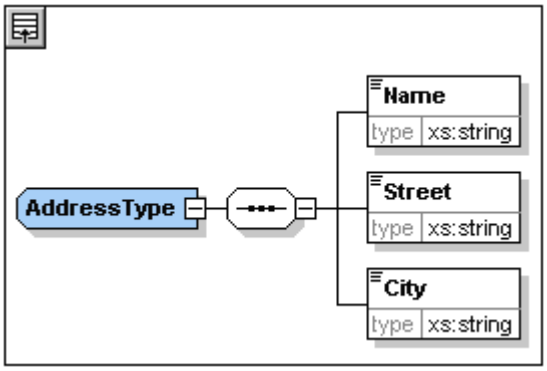


3. Klicken Sie auf das Symbol "Globale Elemente anzeigen" .
   Damit kehren Sie zur Schema-Übersicht zurück, in der nun alle globalen Komponenten angezeigt werden.
4. Klicken Sie in der Komponenten-Eingabehilfe auf das Erweiterungssymbol für **Element-** und **ComplexType**-Einträge, um die jeweiligen Schema-Konstrukte zu sehen.
   In der Schema-Übersicht werden nun zwei globale Komponenten angezeigt: das Element `Company` und der ComplexType `AddressType`. Außerdem wird in der Komponenten-Eingabehilfe der ComplexType `AddressType` angezeigt.

© 2005 Altova GmbH



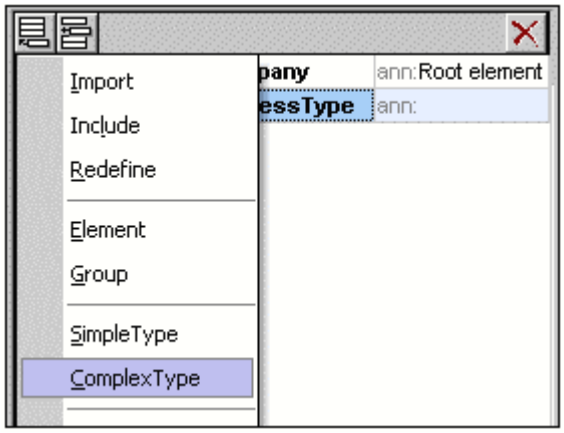
5. Klicken Sie auf das Symbol "Content Model-Ansicht" von `AddressType`, um dessen Content Model anzuzeigen (*siehe Abbildung unten*). Beachten Sie die Form des ComplexType-Behälters.



6. Klicken Sie auf das Symbol "Globale Elemente anzeigen", um zur Schema-Übersicht zurückzukehren.

Erweitern einer "complex type" Definition
Nun werden wir die globale Komponente `AddressType` verwenden, um zwei Arten von länderspezifischen Adressen zu erstellen. Hierzu wird zuerst ein neuer complexType erzeugt, welcher auf der Komponente `AddressType` basiert. Anschließend wird diese Definition erweitert. Gehen Sie dazu folgendermaßen vor:

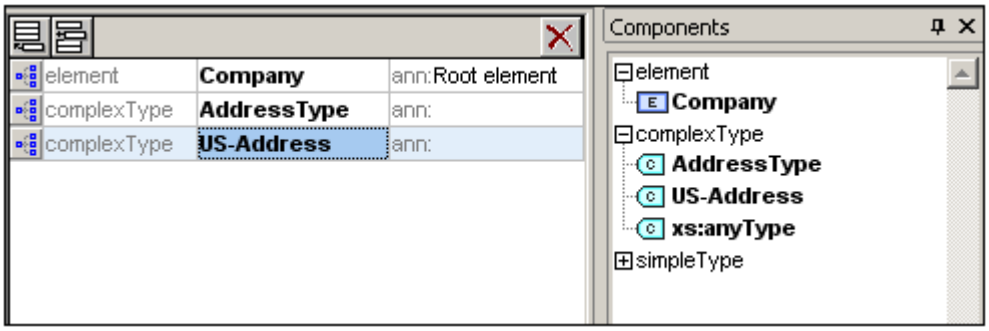
1. Wechseln Sie zur Schema-Übersicht (Wenn Sie sich in der Content Model-Ansicht befinden, klicken Sie auf "Globale Elemente anzeigen").
2. Klicken Sie auf das Symbol **Anhängen** (oben links im Komponentenfenster). Daraufhin wird das folgende Menü angezeigt:



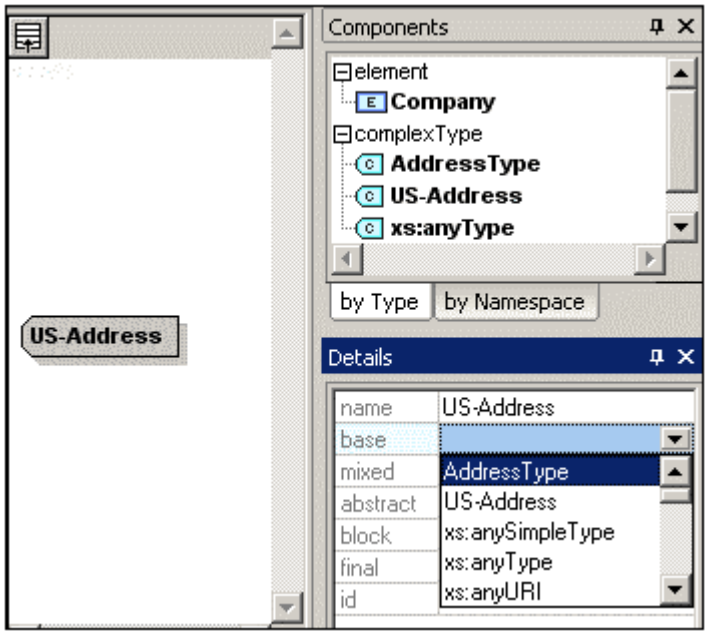
3. Wählen Sie **ComplexType** aus dem Kontextmenü. Eine neue Leerzeile wird in der Komponentenliste angehängt, in der Sie den Namen der Komponente eingeben

können.

4. Geben Sie `US-Address` ein und bestätigen Sie mit der **Eingabetaste**. (Sollten Sie den Bindestrich "-" beim Eingeben vergessen, und stattdessen ein Leerzeichen verwenden, wird der Name in Rot angezeigt. D.h., ein nicht erlaubtes Schriftzeichen wurde verwendet.)

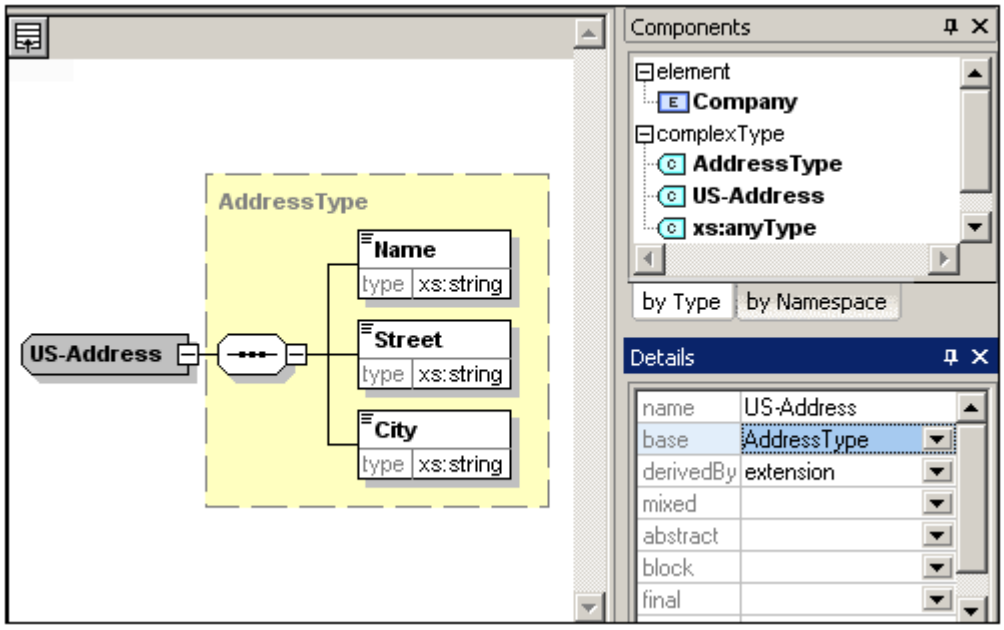


5. Klicken Sie auf das Symbol "Content Model-Ansicht" in der Zeile `US-Address`, um das Content Model des neuen ComplexType zu sehen. Das Content Model ist leer (*siehe Abbildung unten*).
6. Klicken Sie in der Details-Eingabehilfe auf das Listenfeld `base` und wählen Sie `AddressType`.

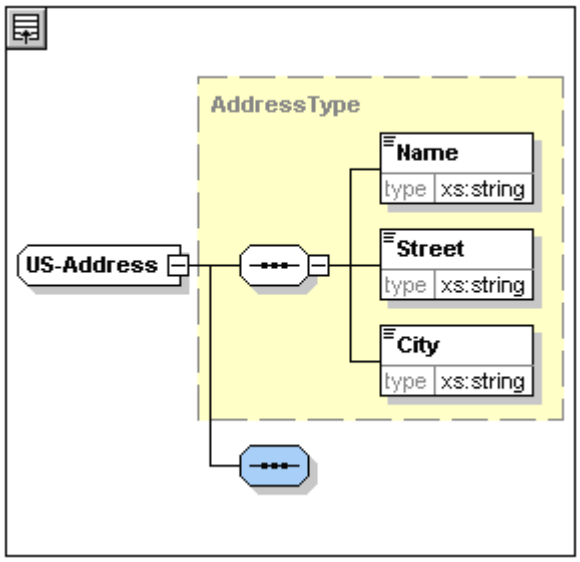


In der Content Model-Ansicht wird nun das Content Model `AddressType` als Content Model von `US-Address` (*siehe Abbildung unten*) angezeigt.

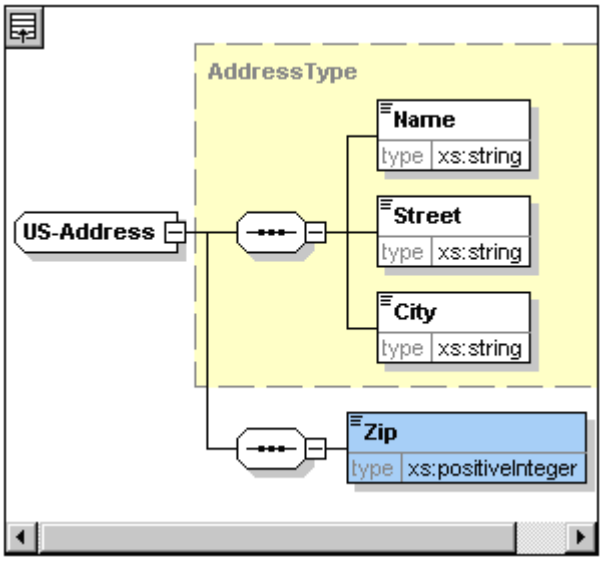
© 2005 Altova GmbH



7. Wir können nun das Content Model des ComplexType `US-Address` um ein ZIP Code-Element erweitern. Klicken Sie dazu mit der rechten Maustaste auf das Element `US-Address` und wählen Sie **Child hinzufügen | Sequence**.
   Ein neuer Sequence-Kompositor wird außerhalb des Kastens `AddressType` (*Abbildung unten*) angezeigt; d.h., es handelt sich um die Erweiterung zu einem Element.



8. Klicken Sie mit der rechten Maustaste auf den neuen **Sequence**-Kompositor und wählen Sie **Child hinzufügen | Element**.
9. Benennen Sie das neu erstellte Element mit `Zip` und drücken Sie die **Tabulatortaste**. Daraufhin wird der Cursor in das Wertefeld der Typbeschreibungszeile gesetzt.
10. Wählen Sie (in der Auswahlliste, die angezeigt wird, wenn Sie in das Wertefeld doppelklicken) `xs:positiveInteger` oder geben Sie den Wert ein und bestätigen Sie mit der **Eingabetaste**.

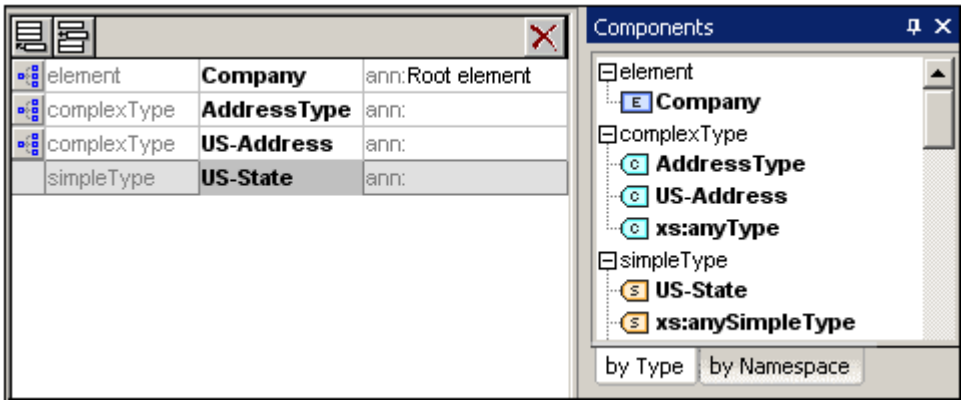


Sie haben nun einen ComplexType namens `US-Address`, der auf dem ComplexType `AddressType` basiert und diesen um einen ZIP Code (Postleitzahl) erweitert.

Globale SimpleTypes
Genau wie der ComplexType `US-Address` auf dem ComplexType `AddressType` basiert, kann ein Element auch auf einem SimpleType basieren. Der Vorteil ist derselbe wie bei globalen ComplexTypes: Der SimpleType kann wiederverwendet werden. Dazu muss der SimpleType zuerst als globales Element definiert werden. In diesem Tutorial werden wir ein Content Model für US-Bundesstaaten in einem SimpleType definieren. Dieser SimpleType wird als Basis für ein weiteres Element verwendet.

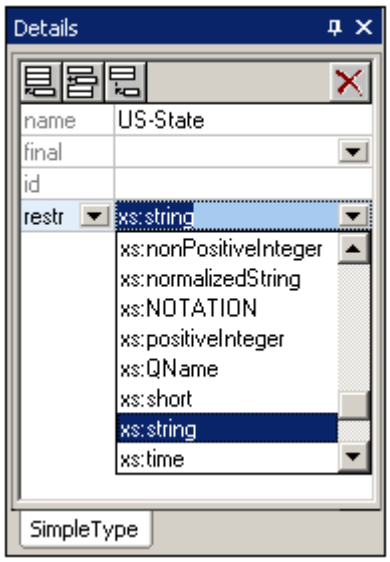
Erstellen eines globalen SimpleType
Um einen globalen SimpleType zu erstellen, müssen Sie einen neuen SimpleType an die Liste der globalen Komponenten anhängen, diesen benennen und seinen Datentyp definieren. Gehen Sie dazu folgendermaßen vor:

1. Wechseln Sie in die Schema-Übersicht (Klicken Sie auf "Globale Elemente anzeigen", wenn Sie sich in der Content Model-Ansicht befinden).
2. Klicken Sie auf das Symbol **Anhängen** und wählen Sie im Kontextmenü Simple Type.
3. Geben Sie `US-State` als Elementname ein.
4. Bestätigen Sie mit der **Eingabetaste**. Der SimpleType `US-State` wird erstellt und in der Komponenten-Eingabehilfe in die Liste der SimpleTypes aufgenommen (Klicken Sie auf das Erweiterungssymbol des SimpleType-Eintrags, um es zu sehen).



 © 2005 Altova GmbH

6. Platzieren Sie in der Details-Eingabehilfe (*Abbildung unten*) den Cursor in das Wertefeld von `restr` und geben Sie `xs:string` ein oder wählen Sie `xs:string` in der Dropdown-Liste im Wertefeld `restr` aus.

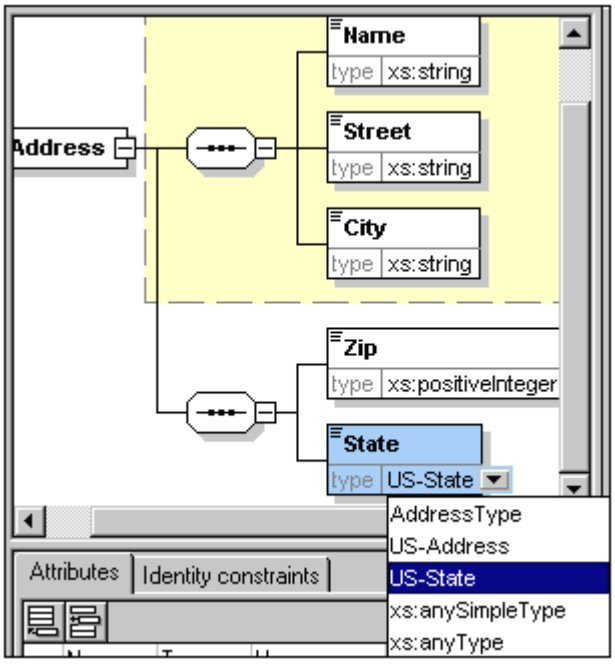


Damit wird ein SimpleType namens `US-State` vom Datentyp `xs:string` erstellt. Diese globale Komponente kann nun im Content Model von `US-Address` verwendet werden.

Verwendung eines globalen SimpleType in einem Content Model

Ein globaler SimpleType kann in einem Content Model dazu verwendet werden, den Typ einer Komponente zu definieren. Wir werden `US-State` verwenden, um im Content Model von `US-Address` ein Element namens `State` zu definieren. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie in der Schema-Übersicht auf das Symbol "Content Model-Ansicht" von `US-Address`.
2. Klicken Sie mit der rechten Maustaste auf den unteren Sequence-Kompositor und wählen Sie **Child hinzufügen | Element**.
3. Geben Sie `State` als Elementnamen ein.
4. Drücken Sie die Tabulatortaste, um den Cursor in das Wertefeld der Typbeschreibungszeile zu setzen.
5. Wählen Sie in der Dropdown-Liste den Eintrag `US-State` aus und bestätigen Sie mit der Eingabetaste.

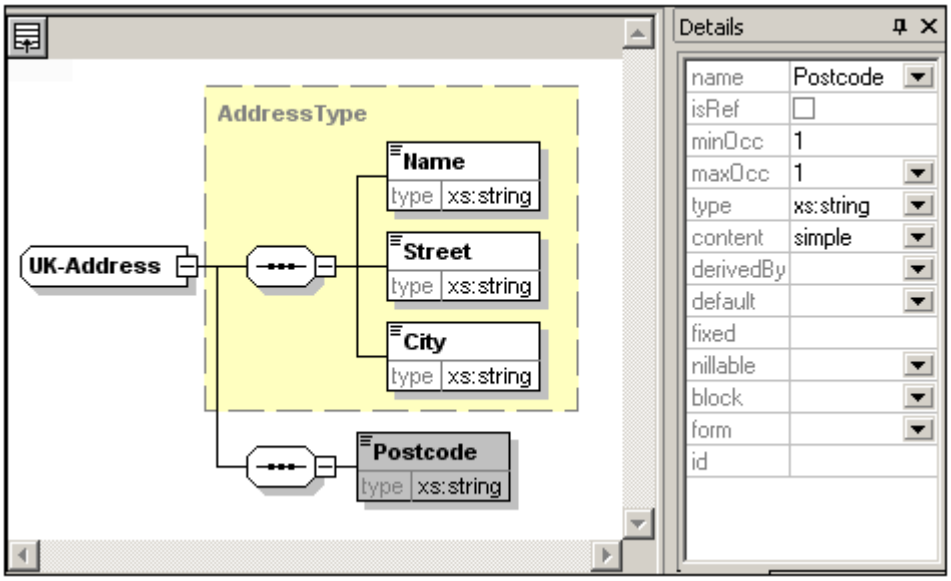


Das Element `State` basiert nun auf dem SimpleType `US-State`.

Erstellen eines zweiten ComplexType auf Basis von `AddressType`
Wir erstellen nun einen globalen ComplexType für Adressen in Großbritannien. Der ComplexType basiert auf dem Element `AddressType` und wird erweitert, bis er dem englischen Adressformat entspricht. Gehen Sie dazu folgendermaßen vor:

1. Erstellen Sie in der Schema-Übersicht auf Basis von `AddressType` einen globalen ComplexType mit dem Namen `UK-Address. (base=AddressType).`
2. Hängen Sie in der Content Model-Ansicht von UK-Address das neue Element `Postcode` an und weisen Sie ihm den Typ `xs:string` zu.

   Das Content Model von `UK-Address` sollte nach erfolgreicher Erstellung folgendermaßen aussehen:



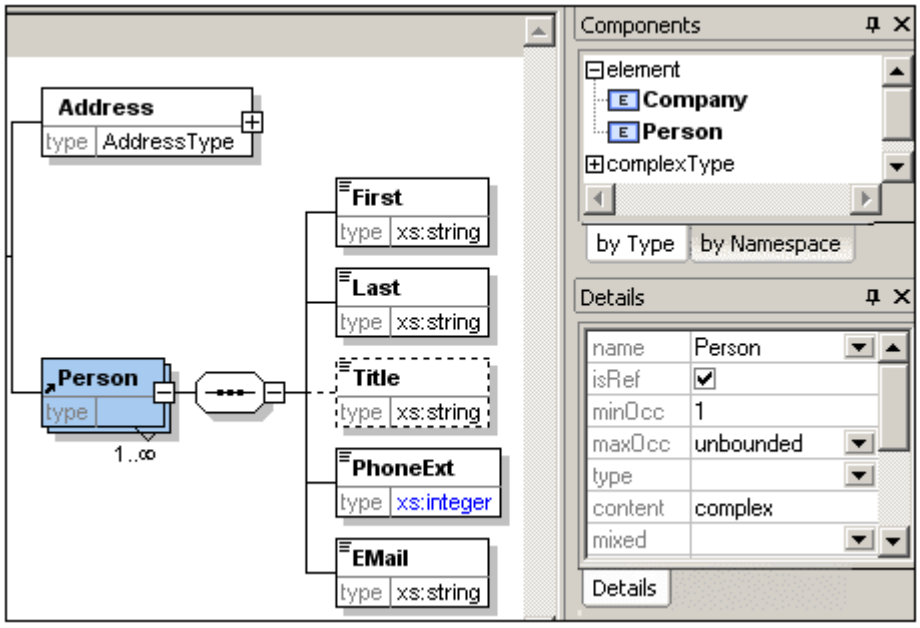
 © 2005 Altova GmbH

Hinweis: In diesem Abschnitt haben Sie globale Simple und ComplexTypes erstellt und anschließend in Content Model-Definitionen verwendet. Der Vorteil von globalen Typen liegt darin, dass diese in mehreren Definitionen wiederverwendet werden können.

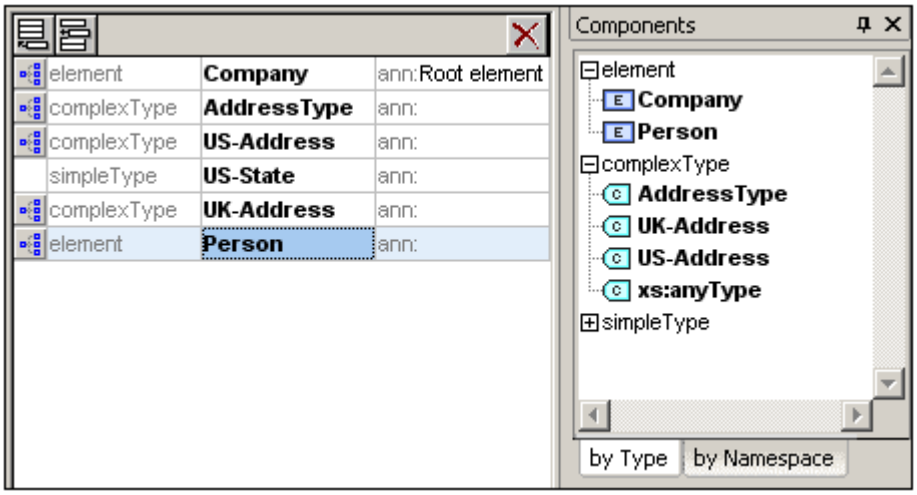
Referenzieren von globalen Elementen

In diesem Abschnitt werden wir das lokal definierte Element `Person` als globales Element definieren und es dann vom Element `Company` aus referenzieren.

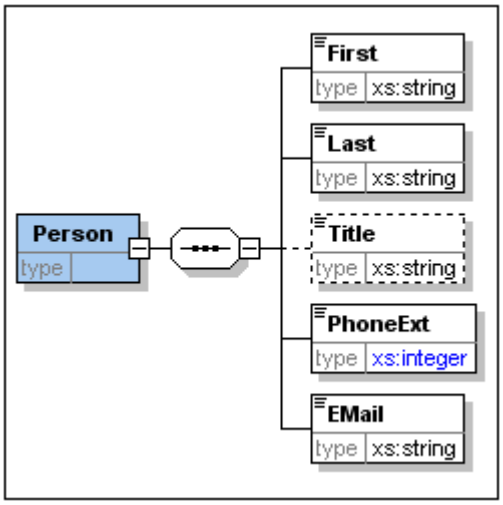
1. Wechseln Sie in die Schema-Übersicht (Globale Elemente anzeigen).
2. Klicken Sie auf das Symbol "Diagramm anzeigen" in der Zeile `Company`**.**
3. Klicken Sie mit der rechten Maustaste auf das Element `Person` und wählen Sie **Als global definieren | Element**.
   Ein kleines "link"-Symbol wird in das Element `Person` eingefügt, das anzeigt, dass das Element das als global deklarierte Element `Person` referenziert. Das Feld `isRef` in der Details Eingabehilfe ist nun auf aktiv gesetzt.



4. Klicken Sie auf das Symbol "Globale Elemente anzeigen" um zur Schema-Übersicht zurückzukehren.
   Das Element `Person` wird nun als globales Element aufgelistet und erscheint auch in der Komponenten-Eingabehilfe.



5. Klicken Sie in der Komponenten-Eingabehilfe auf das Element Person, um das Content Model des globalen Elements Person anzuzeigen.



Bitte beachten Sie, dass das Kästchen für das globale Element kein Link-Symbol hat, da es sich hierbei um das referenzierte Element und nicht das referenzierende Element handelt. Nur das referenzierende Element ist mit einem Link-Symbol gekennzeichnet.

Hinweis:

- Ein Element, das ein globales Element referenziert, muss denselben Namen haben, wie das globale Element, das es referenziert.
- Globale Deklarationen geben keine Auskunft über die Positionierung der Elemente innerhalb des XML-Dokumentes. Sie bestimmen nur den Inhalt des jeweiligen Elementes. Nur wenn eine globale Deklaration von einer anderen Komponente aus referenziert wird, wird ihre Position im XML-Dokument definiert.

  Ein global deklariertes Element kann an mehreren Stellen wiederverwendet werden. Es unterscheidet sich insofern von einem global deklarierten ComplexType, als sein Content Model nicht geändert werden kann, ohne dass auch das globale Element selbst geändert wird. Wenn Sie das Content Model eines Elements ändern, das ein globales Element referenziert, so wird auch das Content Model des globalen Elements geändert und damit auch das Content Model aller anderen Elemente, die das globale Element referenzieren.

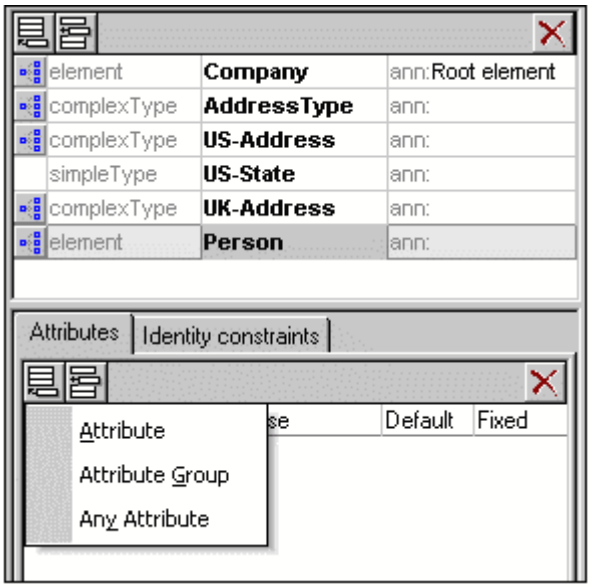
© 2005 Altova GmbH

Attribute und Attribut-Enumerationen

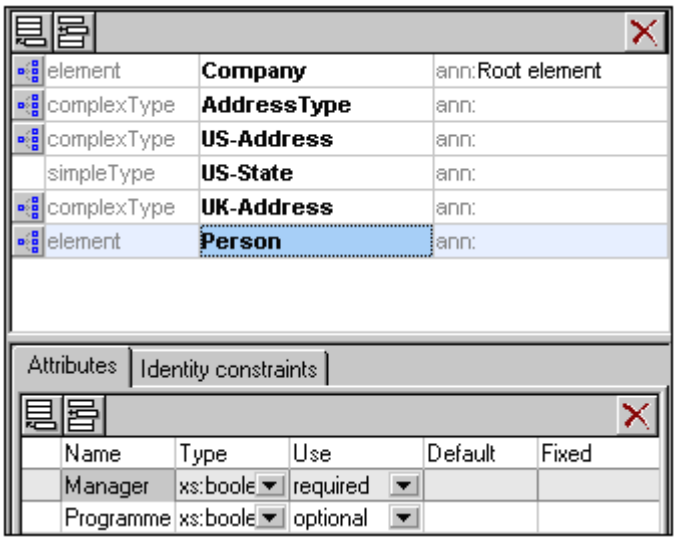
In diesem Abschnitt lernen Sie, wie man Attribute und Enumerationen für Attribute erstellt.

Definieren von Element-Attributen

1. Klicken Sie in der Schema-Übersicht auf das Element `Person` um es aktiv zu setzen.
2. Klicken Sie auf das Symbol "Anhängen", (oben links im Fenster "Attribute/Identity Constraints", das den unteren Bereich des Schema-Übersichtsfensters einnimmt) und wählen Sie den Eintrag "Attribute".



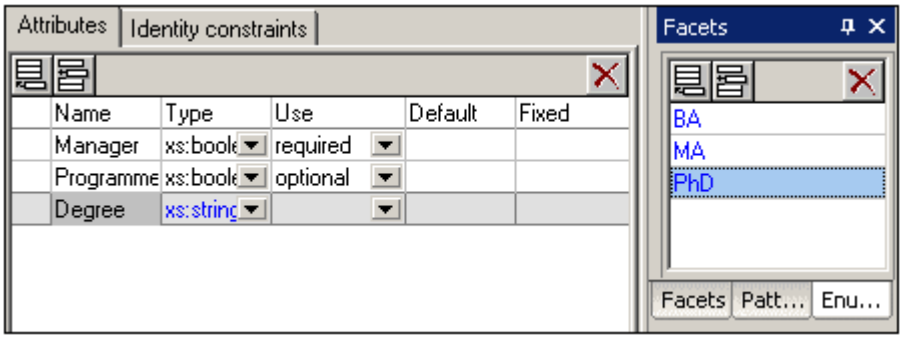
3. Geben Sie `Manager` in das Namensfeld ein.
4. Wählen Sie in der Dropdown-Liste `Type` den Eintrag `xs:boolean`.
5. Wählen Sie in der Dropdown-Liste `Use` den Eintrag `required`.



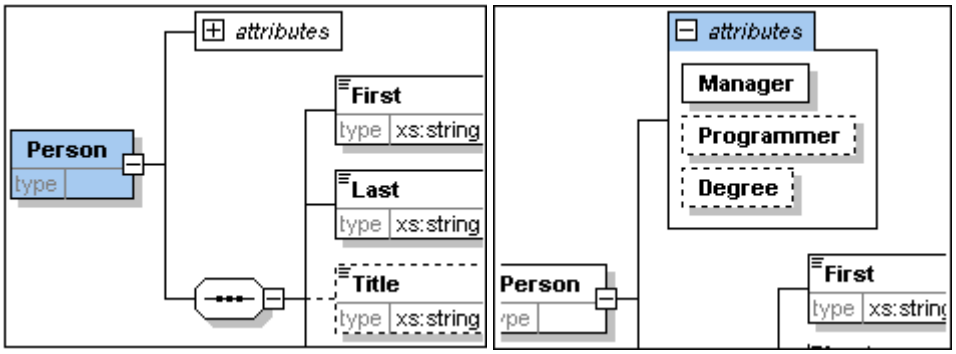
6. Erstellen Sie nun auf dieselbe Art ein Attribut `Programmer` mit einem `Type=xs:boolean` und `Use = optional`.

Definieren von Enumerationen für Attribute
Enumerationen sind Werte, die bei einem bestimmten Attribut zulässig sind. Wenn der Wert des Attributs im XML-Instanzdokument nicht in einer im XML-Schema definierten Enumeration enthalten ist, ist das Dokument ungültig. Wir werden Enumerationen für das Attribut `Degree` des Elements `Person` erstellen. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie auf das **Anhängen-**Symbol oben links im Attribut-Fenster und wählen Sie den Eintrag "**Attribute**".
2. Geben Sie `Degree` als Attributsnamen ein und wählen Sie `xs:string` als Attribut-Type.
3. Klicken Sie in der Facets-Eingabehilfe auf das Register **Enumerations** (*siehe Abbildung unten*).



4. Klicken Sie auf dem Register "Enumerations" auf das Anhängen-Symbol .
5. Geben Sie BA (Bachelor of Arts) ein und bestätigen Sie mit der Eingabetaste.
6. Mit derselben Methode geben Sie zwei weitere Werte in die Enumerations-Liste ein: `MA` und `PhD`.
7. Klicken Sie auf das Symbol "Content Model Ansicht" des Elements `Person`.



Die zuvor definierten Attribute werden in der Content Model-Ansicht angezeigt. Wenn Sie auf das Erweiterungssymbol klicken, werden alle für dieses Element definierten Attribute angezeigt. Sie können diesen Anzeigemodus ein- und ausschalten, indem Sie die Menüoption **Schema-Design | Ansicht konfigurieren** wählen und auf die Kontrollkästchen für Attribute und Identity Constraints klicken.

8. Klicken Sie auf das Symbol "Globale Elemente anzeigen", um zur Schema-Übersicht zurückzukehren.

Speichern des fertigen Schemas
Speichern Sie das fertige Schema unter einem Namen Ihrer Wahl (**Datei | Speichern unter**). Wir empfehlen Ihnen, es unter dem Namen `AddressLast.xsd` zu speichern, da die XML-Datei, die Sie im nächsten Teil des Tutorials erstellen werden, auf dem Schema `AddressLast.xsd` basieren wird. Bevor Sie Ihre Schema-Datei jedoch unter diesem Namen

 © 2005 Altova GmbH

speichern, sollten Sie zuerst die vorhandene Datei `AddressLast.xsd` umbenennen (z.B. in `AddressLast_original.xsd`).

2.1.4 Schemanavigation und Dokumentation

Nach Fertigstellung Ihres XML-Schemas sollten Sie sich nun mit einigen Navigationskürzeln vertraut machen und etwas über die Schemadokumentation, die Sie in XMLSpy 2005 generieren können, erfahren. Diese Themen werden in den folgenden Unterabschnitten erörtert.

In diesem Abschnitt verwendete Befehle
Wir verwenden in diesem Abschnitt des Tutorials ausschließlich die Schema/WSDL-Ansicht. Dabei werden die folgenden Befehle verwendet:

Diagramm anzeigen (oder Content Model-Ansicht anzeigen). Mit diesem Symbol, das in der Schema-Übersicht links von allen globalen Komponenten angezeigt wird, sehen Sie das Content Model der dazugehörigen globalen Komponente.

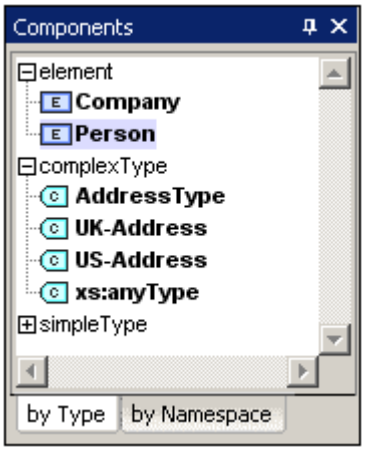
Schemanavigation

In diesem Abschnitt wird beschrieben, wie Sie effizient innerhalb der Schema/WSDL-Ansicht navigieren können. Probieren Sie diese Navigationsverfahren am besten aus, um sich damit vertraut zu machen.

Anzeigen des Content Model einer globalen Komponente
Globale Komponenten, die ein Content Model haben können, sind ComplexTypes, Elemente und Elementgruppen. Die Content Model-Ansicht dieser Komponenten lässt sich auf folgende Arten aufrufen:

- Klicken Sie in der Schema-Übersicht auf das Symbol "Diagramm anzeigen" links vom Namen der Komponente.

- Doppelklicken Sie entweder in der Schema-Übersicht oder in der Content Model-Ansicht in der Komponenten-Eingabehilfe auf das Element, den ComplexType oder die Elementgruppe (*siehe Abbildung unten*).



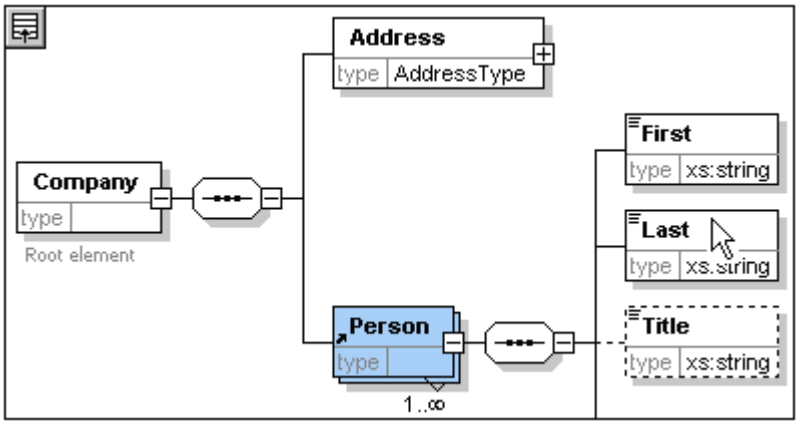
Wenn Sie in der Komponenten-Eingabehilfe auf einer der anderen globalen Komponenten (SimpleType, Attribut, Attributgruppe) doppelklicken, erscheint diese

Komponente in der Schema-Übersicht (da diese Komponenten kein Content Model haben) markiert.

Das Anzeigen der Komponenten durch Doppelklick funktioniert in der Komponenten-Eingabehilfe sowohl auf dem Register "By Type" als auch auf dem Register "By Namespace".

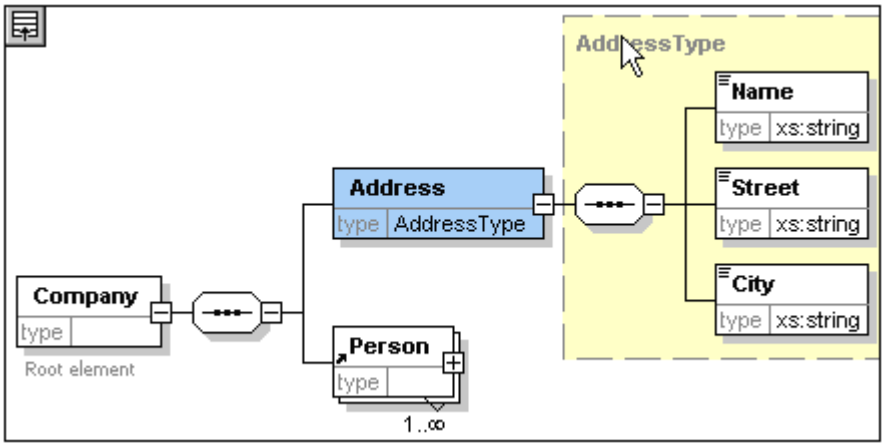
So gelangen Sie von einem referenzierenden Element aus zur Definition eines globalen Elements:
Wenn ein Content Model ein Element enthält, das ein globales Element referenziert, können Sie das Content Model dieses globalen Elements oder das seiner darin enthaltenen Komponenten direkt aufrufen, indem Sie die **Strg**-Taste gedrückt halten, während Sie auf das gewünschte Element doppelklicken.

Wenn Sie z.B. während der Ansicht des Content Model von `Company` die **Strg**-Taste gedrückt halten, während Sie auf `Last` doppelklicken, wird das Content Model `Person` geöffnet und das Element `Last` darin wird markiert.



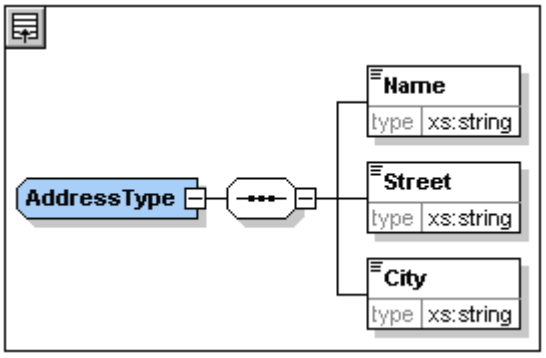
Wenn das Element `Last` markiert ist, werden alle seine Eigenschaften sofort in den entsprechenden Eingabehilfen und Informationsfenstern angezeigt.

So gelangen Sie zur Definition eines ComplexType
ComplexTypes werden oft als Elementtyp in einem Content Model verwendet. Um die Definition eines ComplexType direkt von einem Content Model aus aufzurufen, doppelklicken Sie im gelben Kästchen auf den **Namen** des ComplexType (*siehe Mauszeiger in der Abbildung unten*).



Auf diese Art gelangen Sie zur Content Model-Ansicht des ComplexType.

© 2005 Altova GmbH



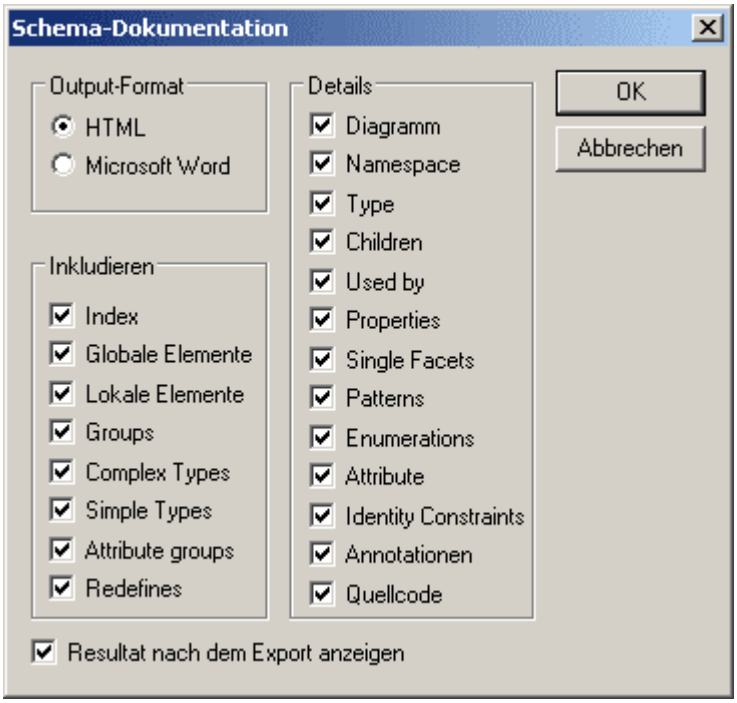
Hinweis: Wie auch bei referenzierten globalen Elementen können Sie direkt zu einem Element innerhalb der ComlexType-Definition gehen, indem Sie die **Strg-**Taste gedrückt halten und auf das gewünschte Element im Content Model, das den ComplexType enthält, doppelklicken.

Schema-Dokumentation

XMLSpy 2005 verfügt über Funktionen zur Erstellung einer detaillierten Dokumentation von XML-Schemas im HTML- und MS Word-Format. Sie können die Komponenten und die Genauigkeit der Dokumentation auswählen. Miteinander in Zusammenhang stehende Komponenten werden im HTML- und im Word-Dokument durch Hyperlinks verbunden. Um Dokumentation im MS Word-Format zu generieren, muss MS Word auf Ihrem Computer (oder Netzwerk) installiert sein.

Sie werden in diesem Abschnitt Dokumentation für das XML-Schema `AddressLast.xsd` generieren. Gehen Sie dazu folgendermaßen vor:

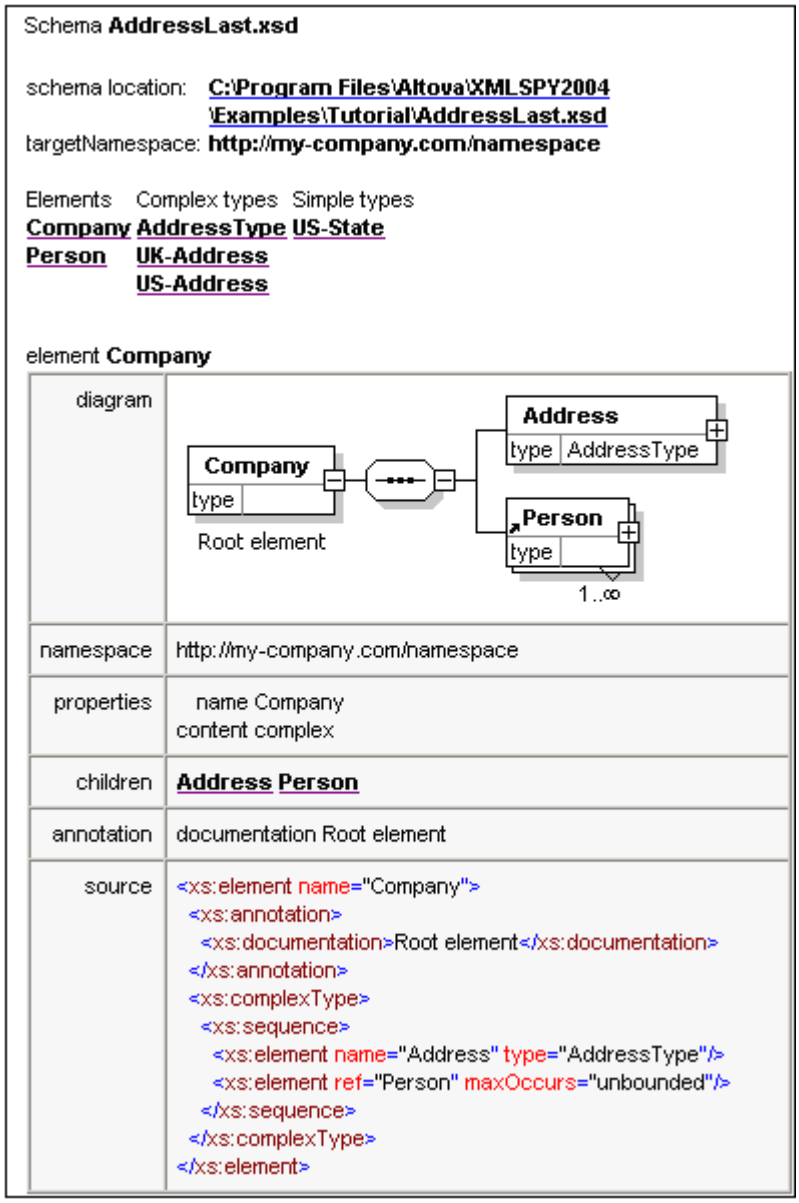
1. Wählen Sie die Menüoption **Schema-Design | Schema-Dokumentation generieren,** um das Dialogfeld "Schema-Dokumentation" aufzurufen.



2. Wählen Sie das Ausgabeformat HTML und bestätigen Sie mit OK.

3. Wählen Sie im Dialogfeld "Speichern unter" das Verzeichnis und geben Sie einen Namen für die HTML-Datei ein (z.B. `AddressLast.html`). Bestätigen Sie mit **Speichern**.

   Das HTML-Dokument wird in der Browser-Ansicht von XMLSpy 2005 angezeigt. Wenn Sie auf einen Link klicken, gelangen Sie zur damit verknüpften Komponente.



Schema **AddressLast.xsd**

schema location: **C:\Program Files\Altova\XMLSPY2004 \Examples\Tutorial\AddressLast.xsd**
targetNamespace: **http://my-company.com/namespace**

Elements	Complex types	Simple types
Company	**AddressType**	**US-State**
Person	**UK-Address**	
	US-Address	

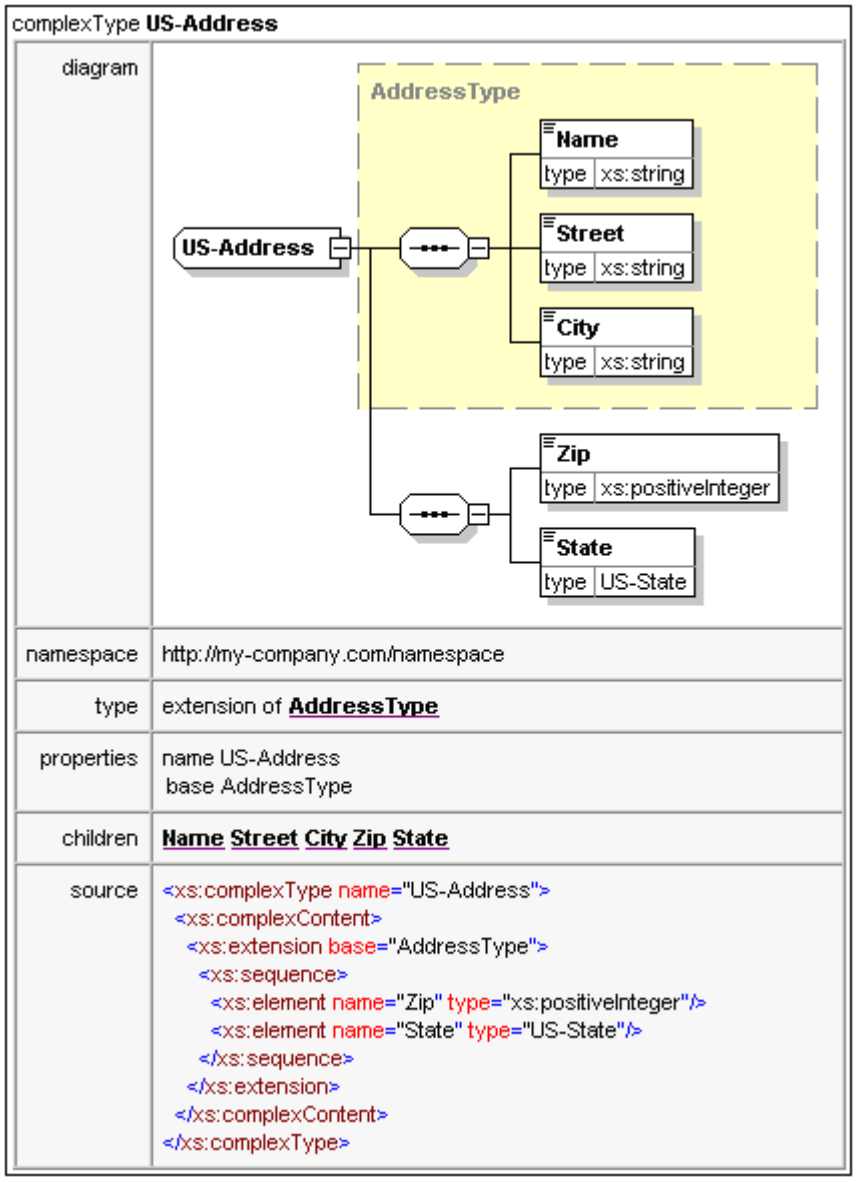
element **Company**

diagram	
namespace	http://my-company.com/namespace
properties	name Company content complex
children	**Address** **Person**
annotation	documentation Root element

source	`<xs:element name="Company">`
	`<xs:annotation>`
	`<xs:documentation>Root element</xs:documentation>`
	`</xs:annotation>`
	`<xs:complexType>`
	`<xs:sequence>`
	`<xs:element name="Address" type="AddressType"/>`
	`<xs:element ref="Person" maxOccurs="unbounded"/>`
	`</xs:sequence>`
	`</xs:complexType>`
	`</xs:element>`

Das oben abgebildete Diagramm stellt die **erste Seite** der Schema-Dokumentation im HTML-Format dar. Falls Komponenten anderer Schemas miteinbezogen wurden, werden diese ebenfalls dokumentiert.

© 2005 Altova GmbH



Das oben abgebildete Diagramm stellt dar, wie ComplexTypes dokumentiert werden.

element **US-Address/Zip**

diagram	**Zip** type xs:positiveInteger
namespace	http://my-company.com/namespace
type	**xs:positiveInteger**
properties	name Zip isRef 0 content simple
source	<xs:element name="Zip" type="xs:positiveInteger"/>

element **US-Address/State**

diagram	**State** type US-State
namespace	http://my-company.com/namespace
type	**US-State**
properties	name State isRef 0 content simple
source	<xs:element name="State" type="US-State"/>

simpleType **US-State**

namespace	http://my-company.com/namespace
type	**xs:string**
properties	name US-State
used by	element **US-Address/State**
source	<xs:simpleType name="US-State"> <xs:restriction base="xs:string"/> </xs:simpleType>

Das oben abgebildete Diagramm zeigt an, wie Elemente und SimpleTypes dokumentiert werden.

Sie können nun die MS Word-Ausgabeoption ausprobieren. Das Word-Dokument wird in MS Word geöffnet. Um Hyperlinks im MS Word-Dokument zu verwenden, halten Sie die **Strg**-Taste gedrückt, während Sie auf den Link klicken.

2.1.5 Erstellen von XML Dokumenten

In diesem Abschnitt lernen Sie:
Erstellen eines neuen XML Dokumentes in XMLSpy 2005, Verwenden von verschiedenen Ansichten und Eingabehilfen für erleichtertes Bearbeiten und Validieren von Daten in XMLSpy 2005. In diesem Abschnitt werden folgende Punkte behandelt:

© 2005 Altova GmbH

- Erstellen eines neuen XML-Dokumentes basierend auf dem Schema `AddressLast.xsd`.
- Definition des Elementtyps, um für das Element bei der Validierung ein erweitertes Content Model für dieses Element bereitzustellen. (Bei Verwendung eines erweiterten Content Model ohne Definition des erweiterten Content Model wird ein Validierungsfehler ausgegeben).
- Einfügen von Elementen und Attributen und Eingabe von Inhalt für diese in der Grid- und der Text-Ansicht mit Hilfe der intelligenten Eingabehilfen.
- Kopieren von XML-Daten von XMLSpy 2005 in ein MS Excel-Blatt, Einfügen von Daten in MS Excel und Zurückkopieren der Daten nach XMLSpy 2005. Diese Funktion steht in der Datenbank/Tabellen-Ansicht der Grid-Ansicht zur Verfügung.
- Sortieren von XML-Elementen mit Hilfe der Sortierfunktion der Datenbank/Tabellen-Ansicht.
- Validieren des XML-Dokumentes.
- Ändern des Schemas**,** so dass dreistellige Telefondurchwahlnummern eingegeben werden können.

In diesem Abschnitt verwendete Befehle

Datei | Neu erstellt eine neue XML-Datei.

Ansicht | Text wechselt in die Text-Ansicht.

Ansicht | Enhanced Grid wechselt in die Enhanced Grid-Ansicht.

XML | Tabelle | Als Tabelle anzeigen zeigt Mehrfachinstanzen eines einzelnen Elementtyps auf einer einzelnen hierarchischen Ebene als Tabelle an. Diese Ansicht des Elements nennt sich Datenbank/Tabellen-Ansicht (oder einfach Tabellenansicht). Das Symbol dient zum Wechseln zwischen der Tabellen,- und der normalen Enhanced Grid-Ansicht.

F7 Zum Kontrollieren, ob das Dokument wohlgeformt ist.

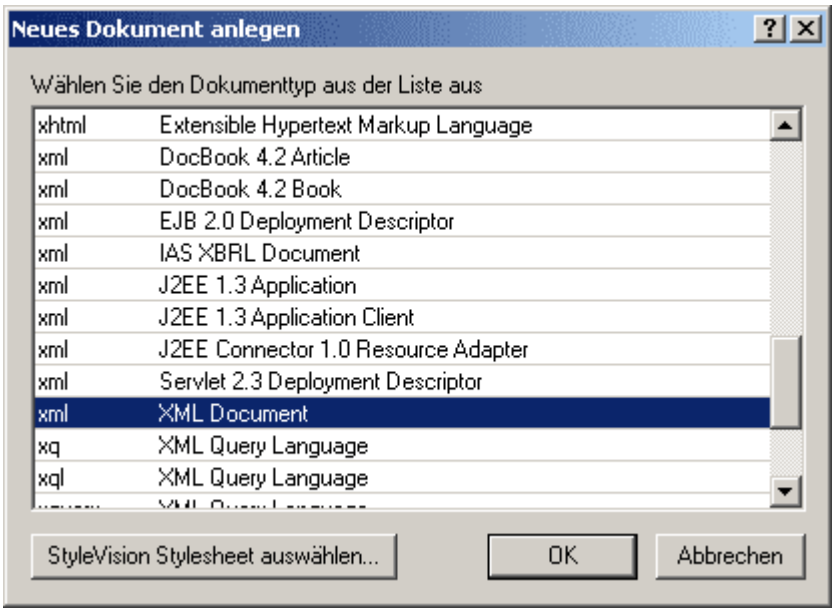
F8 Validiert das XML-Dokument gegen das damit verknüpfte Schema bzw. die DTD.

Öffnet die dazugehörige DTD oder XML-Schema-Datei.

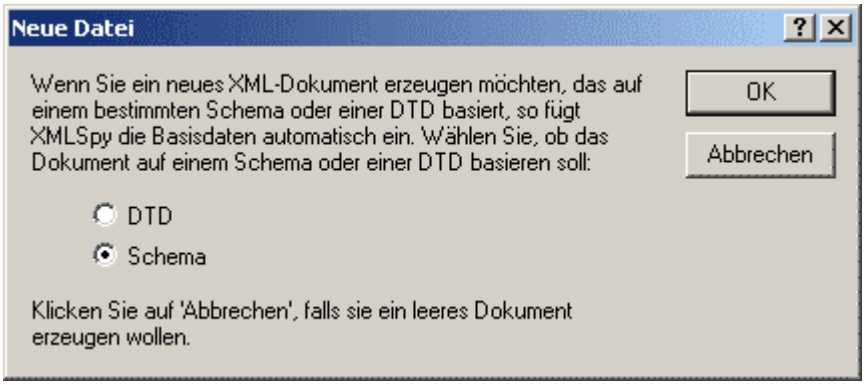
Erstellen eines neuen XML-Dokuments

Beim Erstellen einer neuen XML-Datei in XMLSpy 2005 haben Sie die Option, dieser ein Schema (DTD oder XML-Schema) zugrunde zu legen oder auch nicht. In diesem Abschnitt werden wir eine neue Datei erstellen, die auf dem zuvor im Rahmen dieses Tutorials erzeugten Schema `AddressLast.xsd` basiert. Gehen Sie dazu folgendermaßen vor:

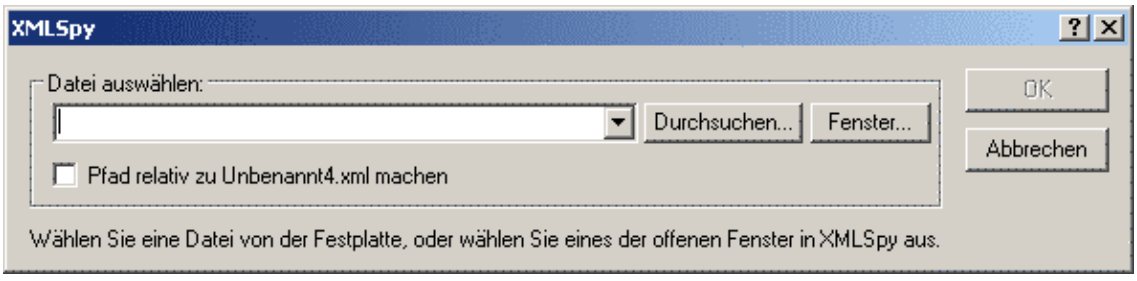
1. Wählen Sie die Menüoption **Datei | Neu**, um das folgende Dialogfeld aufzurufen.



2. Wählen Sie den Eintrag `.xml XML Document` im Dialogfeld und bestätigen Sie mit OK. Im nächsten Fenster werden Sie gefragt, ob das XML-Dokument auf einer DTD oder einem Schema basieren soll.



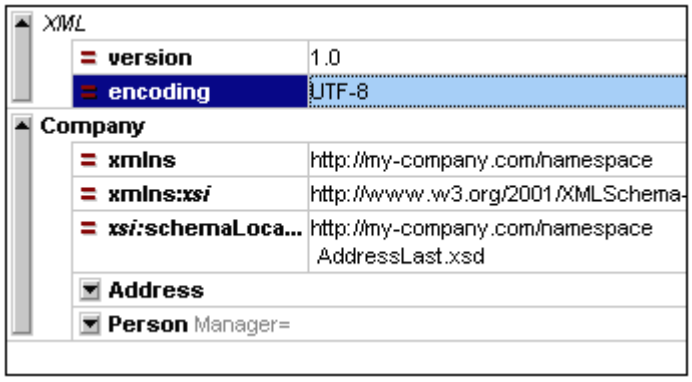
3. Wählen Sie das Optionsfeld "Schema" und bestätigen Sie mit OK. Sie werden nun aufgefordert, die Schema-Datei auszuwählen, auf der Ihr XML-Dokument basieren soll.



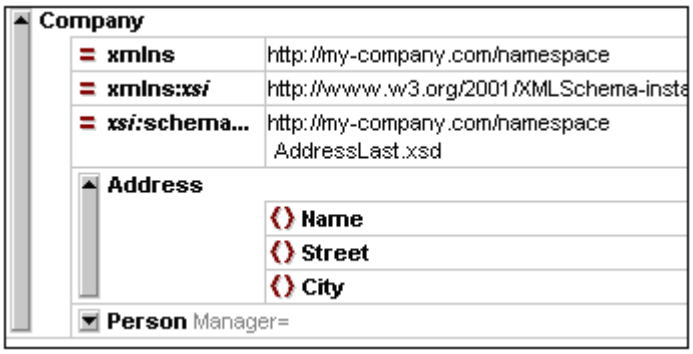
4. Wählen Sie "Durchsuchen" oder "Fenster", um die Schema-Datei zu suchen. Bei Klicken auf die Schaltfläche "Fenster" werden alle Dateien, die in XMLSpy 2005 und Projekten geöffnet sind, aufgelistet. Wählen Sie `AddressLast.xsd` und bestätigen Sie Ihre Auswahl mit OK. Im Hauptfenster wird ein neues XML-Dokument geöffnet. Das Dokument enthält alle Hauptelemente, die im Schema definiert wurden.
5. Wählen Sie die Enhanced Grid-Ansicht aus, um mit dem Tutorial fortzufahren.

 © 2005 Altova GmbH

6. In der Grid-Ansicht ist das gesamte Dokument ausgewählt. Wenn Sie auf ein Element klicken, wird die Auswahl auf dieses Element reduziert. Ihr Dokument sollte etwa folgendermaßen aussehen:

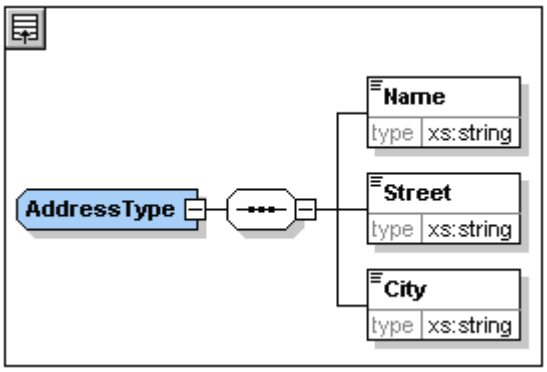


7. Klicken Sie auf das Symbol neben `Address`, um die Child-Elemente von `Address` anzuzeigen. Ihr Dokument sollte aussehen wie dieses:



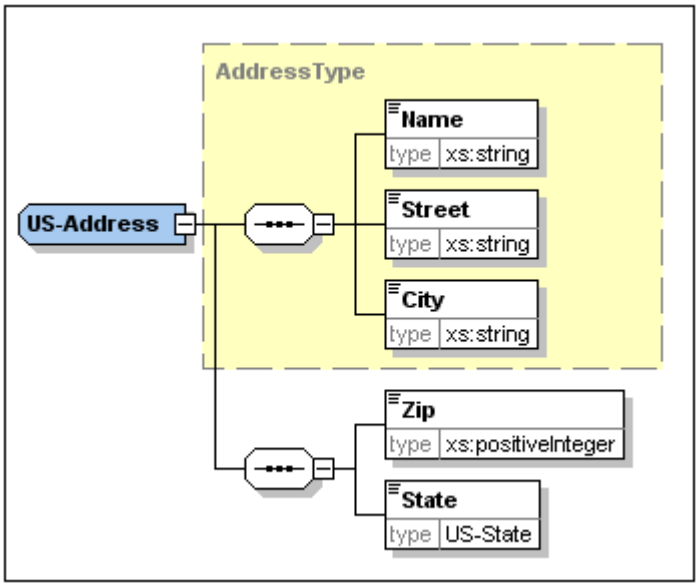
Festlegen des Typs eines Elements

Die in der Grid-Ansicht angezeigten Child-Elemente von `Address` sind die, welche für den globalen ComplexType `AddressType` (*Content Model dazu in der Abbildung unten*) definiert wurden.



Wir möchten jedoch einen spezifischen US- oder UK-Adresstyp verwenden anstelle des generischen Adresstyps. Sie werden sich erinnern, dass wir im Schema `AddressLast.xsd`

globale ComplexTypes für US-Address und UK-Address erstellt haben, indem wir den ComplexType AddressType erweitert haben. Unten sehen Sie das Content Model von US-Address.



Um festzulegen, dass das Element Address im XML-Dokument einem der erweiterten Address-Typen entsprechen muss (US-Address oder UK-Address) und nicht dem generischen AddressType, müssen wir den erforderlichen erweiterten ComplexType als ein Attribut des Elements Address definieren. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf das Element Name und wählen Sie im Kontextmenü **Einfügen | Attribute**.

   Dem Element Address wird ein Attributfeld hinzugefügt.
2. Geben Sie xsi:type als Name des Attributs ein.
3. Drücken Sie die Tabulatortaste um in das nächste (Werte)-Feld zu gelangen. Es erscheint ein Popup-Menü (*siehe unten*), in dem eine Liste der verfügbaren ComplexTypes angezeigt wird.

4. Wählen Sie US-Address aus der Auswahlliste und bestätigen Sie mit der Eingabetaste.

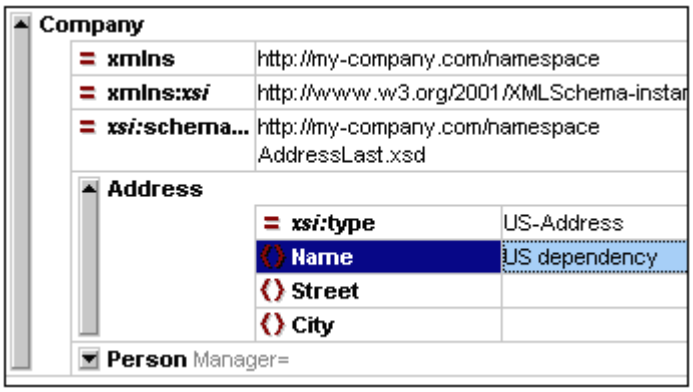
 © 2005 Altova GmbH

Hinweis: Das vorgesetzte `xsi` ermöglicht die Verwendung von speziellen XML-Schema-spezifischen Befehlen im XML-Dokument. Sie haben soeben einen Typ für das Element `Address` festgelegt. Nähere Informationen dazu finden Sie in der XML Schema-Spezifikation.

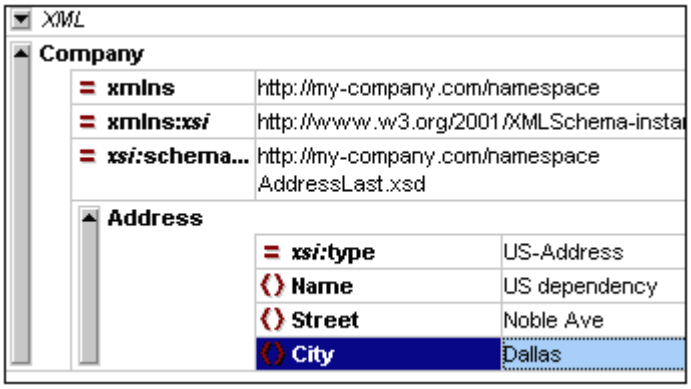
Eingabe von Daten in der Grid-Ansicht

Sie können jetzt Daten in Ihr XML-Dokument eingeben. Gehen Sie dazu folgendermaßen vor:

1. Doppelklicken Sie in das Wertefeld `Name` (oder verwenden Sie die Pfeiltasten) und geben Sie `US dependency` ein. Bestätigen Sie mit der Eingabetaste.



2. Geben Sie auf dieselbe Art einen `Street` und `City` Namen ein (z.B. `Noble Ave.` und `Dallas`).
3. Klicken Sie auf das Element `Person` und drücken Sie die **Entf**-Taste, um es zu löschen. (Wir werden es im nächsten Abschnitt des Tutorials wieder hinzufügen). Dadurch erscheint nun das gesamte Element `Address` markiert.
4. Klicken Sie auf ein beliebiges Child-Element des Elements `Address`, um die Auswahl für alle Child-Elemente von `Address` mit Ausnahme des gewählten Elements aufzuheben. Ihr XML-Dokument sollte folgendermaßen aussehen:



Eingabe von Daten in der Textansicht

Die Textansicht eignet sich aufgrund ihrer intelligenten Editierfunktionen für DTD/XML-Schema ideal zum Editieren der eigentlichen Daten und des Markup-Codes von XML-Dateien.

Funktionen zum Bearbeiten der Struktur

Die Textansicht bietet eine Reihe von Funktionen zur Bearbeitung der Struktur, die das Editieren größerer Textabschnitte vereinfachen. Dazu zählen die folgenden Funktionen, die Sie durch Anklicken des entsprechenden Symbols aktivieren bzw. deaktivieren können.

Aktiviert/deaktiviert die Zeilennummerierung

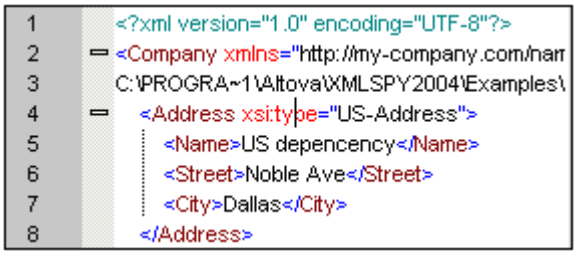
Aktiviert/deaktiviert die Klappleiste

Aktiviert/deaktiviert die Lesezeichenfunktion

Fügt Lesezeichen ein und entfernt diese

Aktiviert/deaktiviert die Einrücklinien

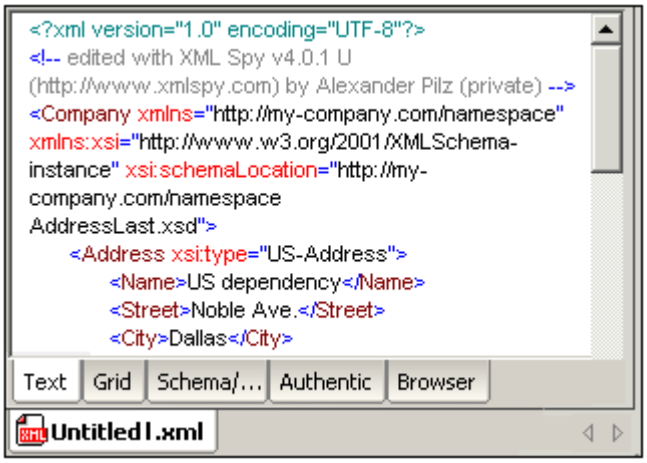
In der unten gezeigten Abbildung sehen Sie die Textansicht der XML-Datei, die sich in Bearbeitung befindet. Alle Sonderfunktionen der Textansicht sind aktiviert. Aus Gründen der Übersichtlichkeit werden diese Funktionen (Zeilennummerierung, Einrücklinien usw.) in weiterer Folge in dieser Anleitung in Textansichten nicht mehr angezeigt. Nähere Informationen zur Textansicht finden Sie im Benutzerhandbuch.



 © 2005 Altova GmbH

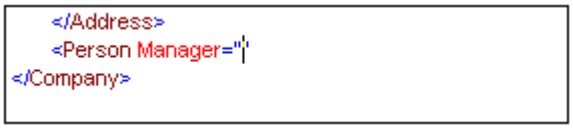
Bearbeitung in der Textansicht
In diesem Abschnitt werden wir Daten in der Textansicht eingeben und bearbeiten, um die Funktionen der Textansicht zu erläutern. Gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Ansicht | Text** oder klicken Sie auf das Register Text. Sie sehen nun das XML-Dokument in seiner Textform mit der Syntaxfärbung.

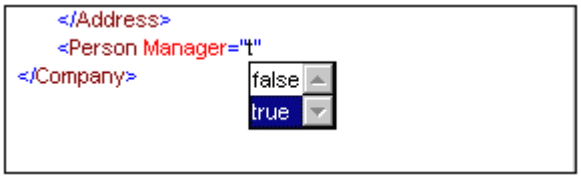


2. Setzen Sie den Cursor hinter den End-Tag des Elements "Address" und drücken Sie die Eingabetaste um eine neue Zeile einzufügen.
3. Geben Sie ein "Kleiner als..." Zeichen (<) an dieser Stelle ein. Eine Auswahlliste aller an dieser Stelle (laut Schema) zulässigen Elemente erscheint. Da an dieser Stelle nur das Element `Person` zulässig ist, wird nur dieses eine Element in der Liste angezeigt.

4. Wählen Sie den Eintrag Person aus. Das Element `Person` wird mit seinem Attribut `Manager` eingefügt.



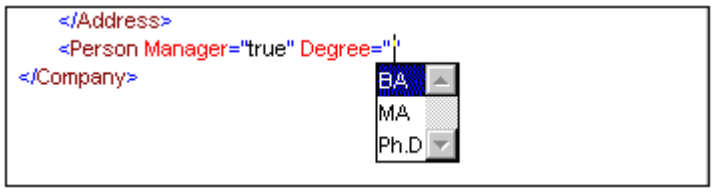
5. Geben Sie den Buchstaben "`t`" innerhalb der Anführungszeichen ein. Daraufhin wird eine Dropdown-Liste angezeigt, in der `true` markiert ist.



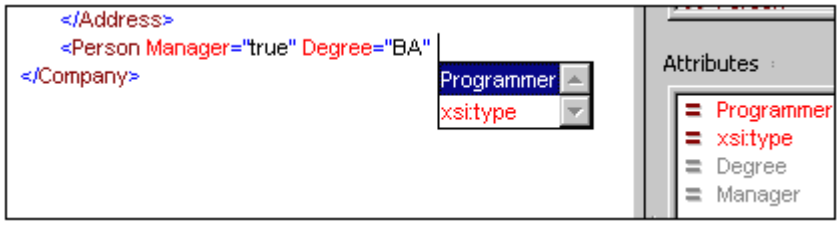
Nach Betätigen der Eingabetaste wird der Wert `true` an der Cursorposition eingefügt.
6. Setzen Sie den Cursor an das Ende der Zeile (oder betätigen Sie die "Ende"-Taste Ihrer Tastatur). Drücken Sie die Leerzeichentaste einmal. Daraufhin wird eine Dropdown-Liste geöffnet, die diesmal eine Liste von an dieser Stelle zulässigen Attributen enthält.

In der Attribut-Eingabehilfe sind verfügbare Attribute rot markiert. Das Attribut "Manager" ist ausgegraut, da es bereits verwendet wurde.

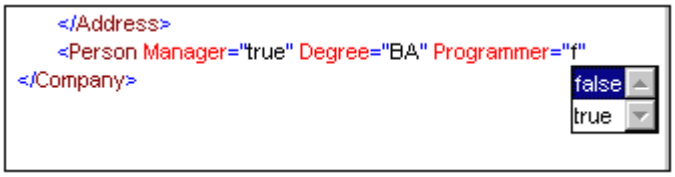
7. Wählen Sie `Degree` durch Betätigen des 'Nach unten'-Pfeiles und drücken Sie die Eingabetaste. Eine weitere Liste wird geöffnet. Hier können Sie eine der vordefinierten Enumerationen auswählen (`BA`, `MA` oder `PhD`).



8. Wählen Sie `BA` aus der Liste, drücken Sie die Eingabetaste und setzen Sie den Cursor durch Drücken der Ende-Taste an das Ende der Zeile. Drücken Sie die Leerzeichentaste. Nun sind `Manager` und `Degree` in der Attribut-Eingabehilfe ausgegraut.



9. Wählen Sie `Programmer` aus der Liste und drücken Sie die Eingabetaste.



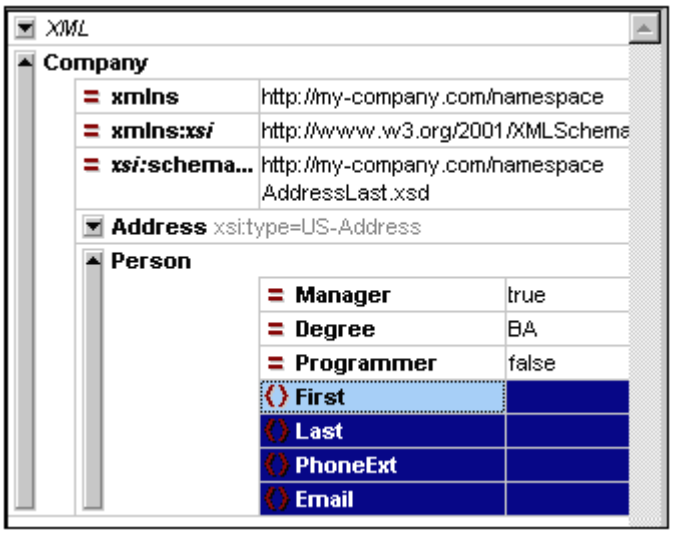
10. Geben Sie den Buchstaben **`"f"`** ein und drücken Sie die Eingabetaste.
11. Setzen Sie den Cursor ans Ende der Zeile, und geben Sie das "größer als..." Zeichen (>) ein. XMLSpy 2005 fügt automatisch alle erforderlichen Child-Elemente von `Person` ein. (Beachten Sie, dass das optionale Element `Title` nicht eingefügt wird). Jedes Element hat einen Start- und einen End-Tag aber keinen Inhalt.

© 2005 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<Company xmlns="http://my-company.com/namespace"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://my-company.com/namespace
AddressLast.xsd">
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave</Street>
        <City>Dallas</City>
    </Address>
    <Person Manager="true" Degree="BA" Programmer="false">
        <First></First>
        <Last></Last>
        <PhoneExt></PhoneExt>
        <Email></Email>
    </Person>
</Company>
```

Sie könnten nun die Personendaten hier eingeben, doch wollen wir nun in die Grid-Ansicht wechseln, damit Sie sehen, wie einfach sich Ansichten beim Bearbeiten eines Dokuments wechseln lassen.

Wechseln zur Enhanced Grid-Ansicht
Um zur Grid-Ansicht zu wechseln, wählen Sie die Menüoption **Ansicht | Enhanced Grid** oder klicken Sie auf das Register Grid. Die neu hinzugefügten Child-Elemente von `Person` erscheinen markiert.



Wir wollen das Dokument nun validieren und etwaige Fehler korrigieren.

Validierung des Dokuments

XMLSpy 2005 bittet zwei Funktionen zur Überprüfung des XML-Dokuments:

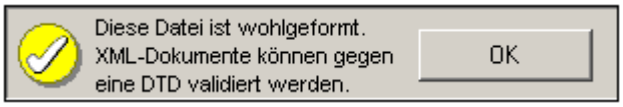
- Wohlgeformtheitprüfung
- Validierung

Wenn ein Fehler bei einer dieser beiden Überprüfungen gefunden wird, muss das Dokument

entsprechend geändert werden.

Überprüfung der Wohlgeformtheit
Ein XML-Dokument ist wohlgeformt, wenn es zu jedem Start-Tag einen End-Tag gibt, Elemente korrekt verschachtelt sind und keine Zeichen fehlen oder an der falschen Stelle stehen (z.B. eine Entity ohne das Semikolon usw.).

Wählen Sie zum Prüfen der Wohlgeformtheit die Menüoption **XML | Wohlgeformtheit prüfen** oder drücken Sie die Taste **F7** oder klicken Sie auf . Am unteren Rand des Hauptfensters erhalten Sie eine Meldung, welche anzeigt, ob das Dokument wohlgeformt ist.

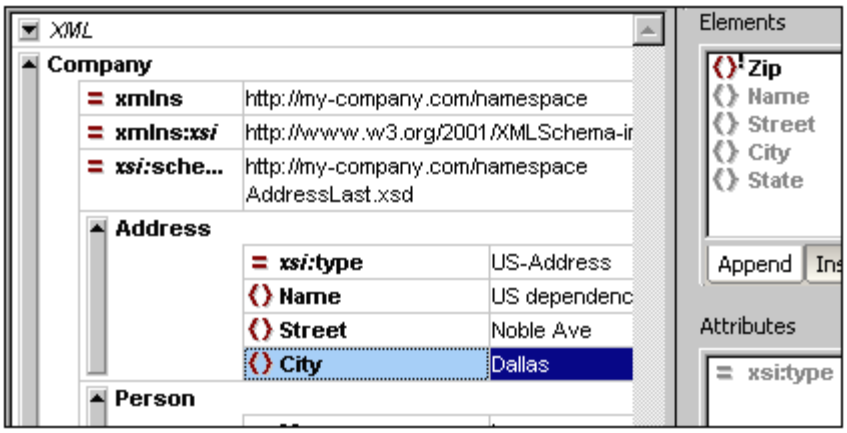


Klicken Sie auf OK, um die Meldung zu schließen. Beachten Sie, dass bei der Wohlgeformtheitsprüfung nicht geprüft wird, ob die Struktur der XML-Datei schemakonform ist. Dies wird bei der Validierung der Datei überprüft.

Überprüfung der Gültigkeit (Validierung)
Ein XML-Dokument ist gemäß einem Schema gültig, wenn es von Struktur und Inhalt mit den Schema-Spezifikationen übereinstimmt.

Wählen Sie die Menüoption **XML | Validieren** oder drücken Sie die Taste **F8** oder klicken Sie auf . Im unteren Bereich des Hauptfensters wird eine Fehlermeldung angezeigt, dass die Datei nicht gültig ist. Nach dem Element `City` in `Address` werden obligatorische Elemente erwartet. Wenn Sie Ihr Schema überprüfen, werden Sie feststellen, dass der ComplexType `US-Address` (dem aufgrund von `xsi:type` attribute dieses `Address`-Element entsprechen muss) ein Content Model hat, in dem auf das Element `City` ein Element `Zip` und ein Element `State` folgen muss.

Beheben von Validierungsfehlern
Das Element, welches den Fehler verursacht hat, wird angezeigt (in diesem Fall "City").

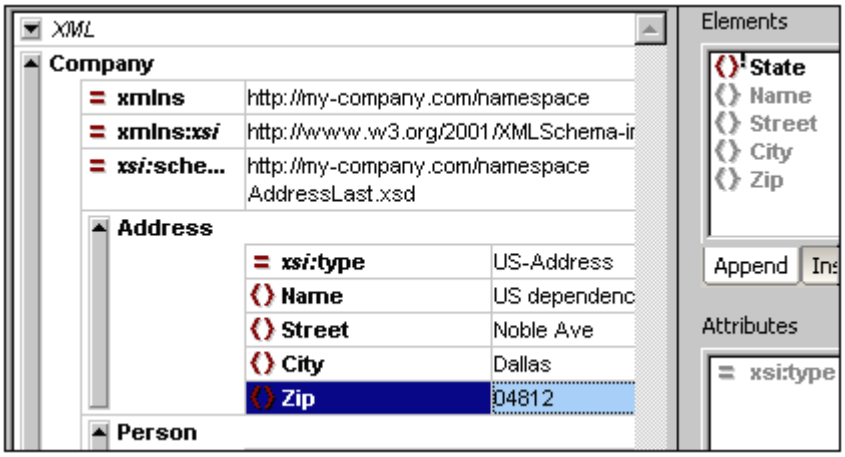


Bitte beachten Sie hierzu auch die Eingabehilfe (oben rechts), bei welcher ein Ausrufungszeichen "!" vor dem Element `Zip` anzeigt, dass es sich hierbei um ein **obligatorisches** Element handelt.

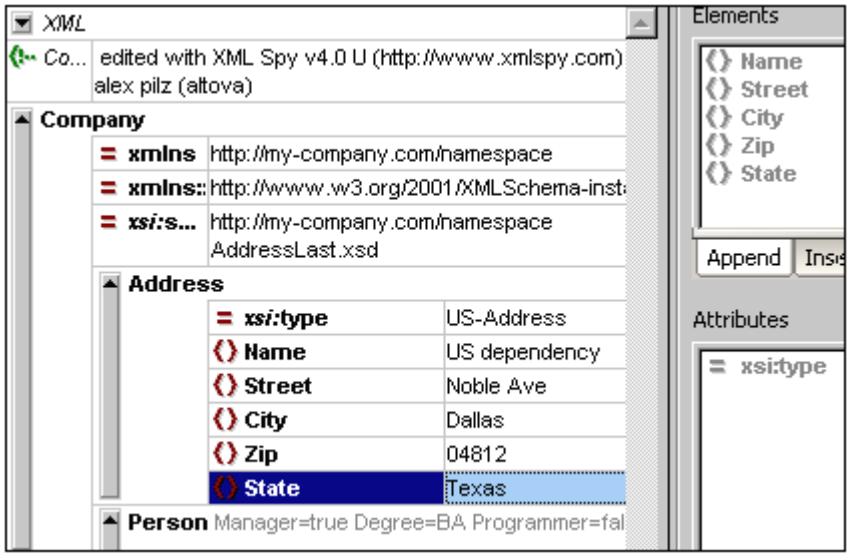
Um den Validierungsfehler zu beheben, gehen Sie folgendermaßen vor:

© 2005 Altova GmbH

1. Doppelklicken Sie auf das Element `Zip` in der Element-Eingabehilfe. Hierdurch wird das Element `Zip` unter dem Element `City` eingefügt. (Das Register "Anhängen" ist in der Element-Eingabehilfe aktiv.)
2. Drücken Sie die Tabulatortaste, geben Sie eine Postleitzahl (Zip) ein - (z.B. `04812`) und bestätigen Sie mit der Eingabetaste.
   Die Element-Eingabehilfe zeigt nun an, dass das Element `State` obligatorisch ist (ein Ausrufezeichen ist dem Element vorangestellt). *Siehe Screenshot unten*.

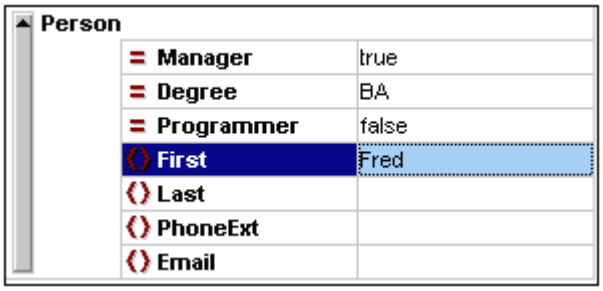


3. Doppelklicken Sie in der Element-Eingabehilfe auf das Element `State`, drücken Sie die Tabulatortaste und geben Sie den Namen des Bundesstaates ein - z.B. `Texas`. Bestätigen Sie mit der Eingabetaste.
   Die Element-Eingabehilfe enthält nun nur noch ausgegraute Elemente. Hierdurch wird angezeigt, dass alle obligatorischen Child-Elemente von `Address` eingefügt worden sind.

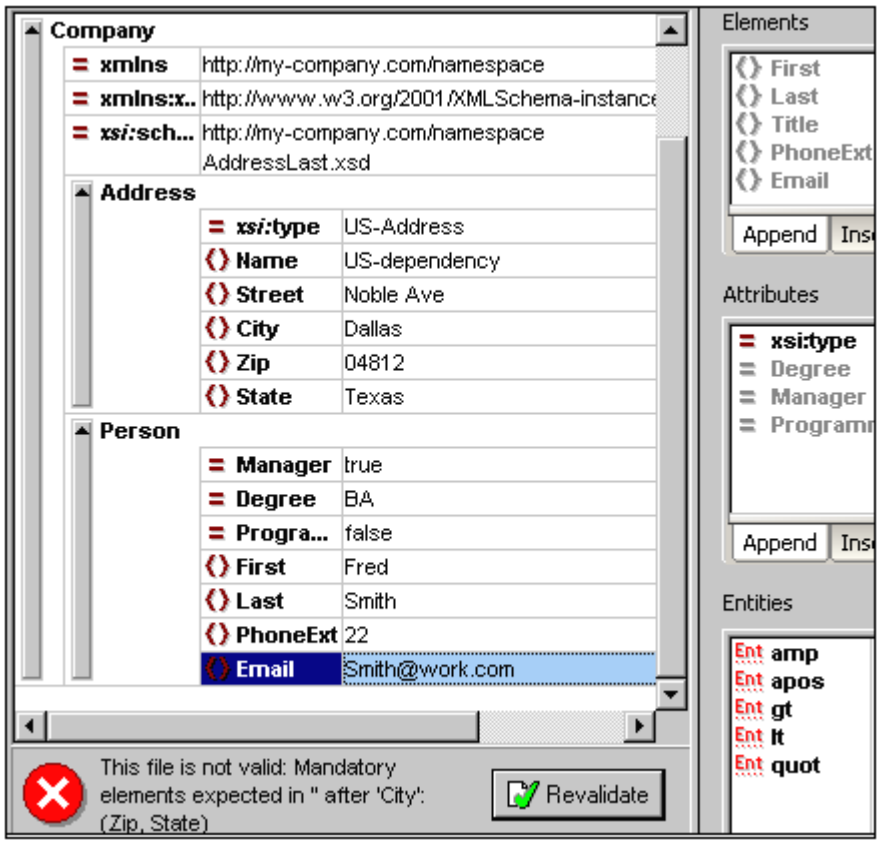


Fertigstellen des Dokuments und erneutes Validieren
Stellen Sie nun das Dokument fertig (geben Sie die Daten für das Element `Person` ein), bevor Sie es erneut validieren. Gehen Sie folgendermaßen vor:

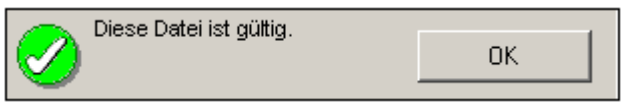
1. Klicken Sie auf das Wertefeld des Elements `First` und geben Sie den Vornamen der Person ein - z.B. `Fred`, bestätigen Sie mit der Eingabetaste.



2. Geben Sie nun auf dieselbe Art Daten für alle Child-Elemente, also `Last`, `PhoneExt` und `Email` ein. Beachten Sie, dass der Wert von `PhoneExt` eine Ganzzahl mit einem Maximalwert von 99 sein muss (da dies der gültige Wertebereich ist, den Sie im Schema definiert haben). Ihr XML-Dokument sollte nun so aussehen:



3. Klicken Sie auf die Schaltfläche **Revalidate,** um das Dokument neuerlich zu prüfen. Diesmal erhalten Sie die Nachricht "`Diese Datei ist gültig`".



4. Das XML Dokument ist nun gegen das Schema validiert und korrekt. Klicken Sie auf OK zum Bestätigen und Schließen der Nachricht.

5. Wählen Sie die Menüoption **Datei | Speichern** und benennen Sie das Dokument - z.B. `CompanyFirst.xml`.Beachten Sie, dass es im Ordner Tutorial bereits eine fertige Datei mit dem Namen `CompanyFirst.xml` gibt, d.h. Sie sollten diese vorher umbenennen.

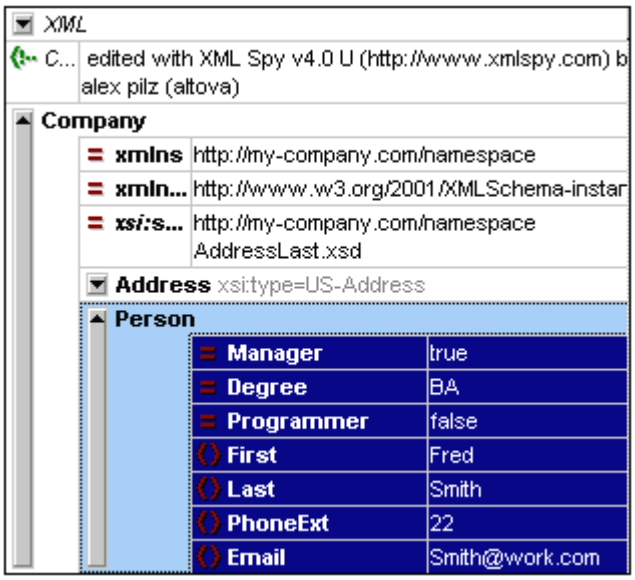
 © 2005 Altova GmbH

Hinweis: Ein XML-Dokument kann auch abgespeichert werden wenn es nicht gültig ist. Beim Speichern werden Sie gefragt, ob Sie das nicht korrekte Dokument speichern wollen. Mit **Save anyway** wird das Dokument in seinem aktuellen, ungültigen Status gespeichert.

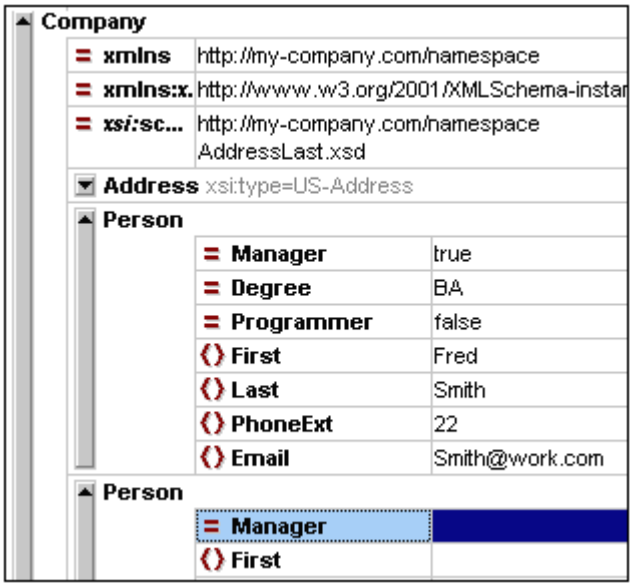
Anhängen von Elementen und Attributen in der Grid-Ansicht

Momentan gibt es im Dokument nur ein `Person`-Element. Um ein neues `Person`-Element hinzuzufügen, gehen Sie folgendermaßen vor:

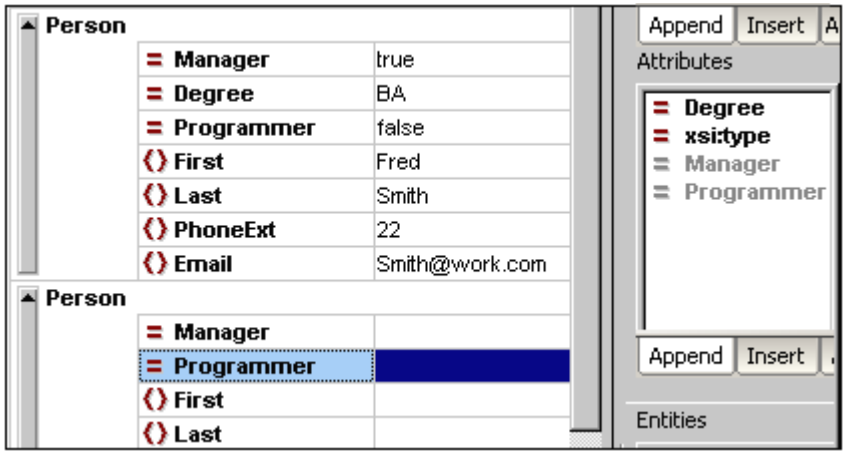
1. Klicken Sie auf die graue Bildlaufleiste links neben dem Element `Address`, um die `Address`-Elemente auszublenden und mehr Platz in der Ansicht zu schaffen.
2. Wählen Sie das gesamte Element Person, indem Sie in der Grid-Ansicht auf oder unter den Text des Elements `Person` klicken. Beachten Sie, dass das Element `Person` nun in der Element-Eingabehilfe auf dem Register "Anhängen" verfügbar ist.



3. Doppelklicken Sie in der Element-Eingabehilfe auf das Element `Person`**.** Ein neues `Person`-Element sowie alle obligatorischen Child-Elemente werden angehängt (*siehe Abbildung unten*). Beachten Sie, dass das optionale Child-Element `Title` nicht eingefügt wird.



4. Klicken Sie auf das Attribut `Manager` im neuen Element `Person`. Beachten Sie die vorhandenen Attribute in der Attribut-Eingabehilfe. Der Eintrag Manager ist ausgegraut, da er bereits verwendet wurde. Werfen Sie auch einen Blick auf das Info-Fenster, in dem nun Informationen über das Attribut `Manager` angezeigt werden.
5. Doppelklicken Sie in der Attribut-Eingabehilfe auf dem Register "Anhängen" auf den Eintrag `Programmer`.
   Hierdurch wird ein leeres `Programmer`-Attribut hinter dem `Manager`-Attribut eingefügt.



Das Attribut `Programmer` ist nun in der Attribut-Eingabehilfe ausgegraut.

Sie könnten in dieser Ansicht einen Inhalt für das Element `Person` eingeben. Wechseln wir jedoch nun in die Datenbank/Tabellen-Ansicht der Grid-Ansicht, da sich diese besser zum Editieren einer Struktur mit mehreren Instanzen wie z.B. bei `Person` eignet.

Editieren in der Datenbank/Tabellen-Ansicht

Die Grid-Ansicht enthält eine spezielle Ansicht namens Datenbank/Tabellen-Ansicht (in weiterer Folge Tabellenansicht genannt), die sich zum Bearbeiten von Elementen mit mehreren Instanzen eignet. Einzelne Elementtypen können als Tabelle angezeigt werden. Wenn ein Elementtyp als Tabelle angezeigt wird, werden dessen Children (Attribute und Elemente) als Spalten dargestellt, wobei die einzelnen Instanzen die Zeilen der Tabelle bilden.

© 2005 Altova GmbH

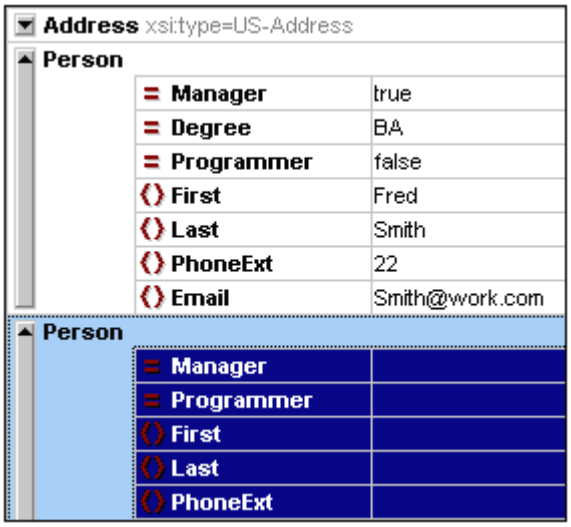
Um einen Elementtyp als Tabelle anzuzeigen, wählen Sie eine beliebige Instanz dieses Elementtyps aus und klicken Sie in der Symbolleiste auf das Symbol "Als Tabelle anzeigen" **(XML | Tabelle | Als Tabelle anzeigen**). Elementtypen einer untergeordneten Ebene, die mehrere Instanzen haben, werden ebenfalls in Tabellenform angezeigt. Die Tabellenansicht steht in der Enhanced Grid-Ansicht zur Verfügung und dient zum Bearbeiten jeder Art von XML-Datei (XML, XSD, XSL, usw.).

Vorteile der Tabellenansicht
Die Tabellenansicht bietet die folgenden Vorteile:

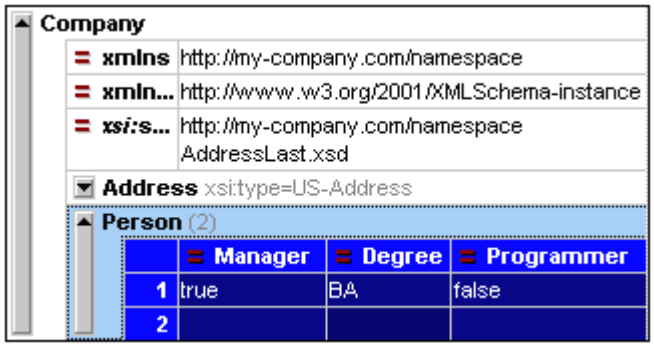
- Sie können Spalten mit Drag & Drop untereinander austauschen. Im XML-Dokument selbst wird dadurch die relative Position von Child-Elementen oder Attributen für alle Element-Instanzen, die den Zeilen der Tabelle entsprechen, geändert.
- Sortieren von Daten nach Spalten mittels der Funktion **XML | Tabelle | Aufsteigend Sortieren** oder **Absteigend sortieren.**
- Einfügen oder Anhängen von Zeilen mittels der Funktion **XML | Tabelle | Zeile Einfügen**
- Kopieren und Einfügen von **strukturierten Daten** von und in Produkte von Drittanbietern
- Auch die intelligenten Eingabehilfen stehen in der Tabellenansicht zur Verfügung.

Anzeigen eines Elementtyps als Tabelle
Um das Element `Person` in Tabellenform anzuzeigen, gehen Sie folgendermaßen vor:

1. Wählen Sie in der Grid-Ansicht eines der `Person`-Elemente aus, indem Sie auf oder in die Nähe des `Person`-Texts klicken.



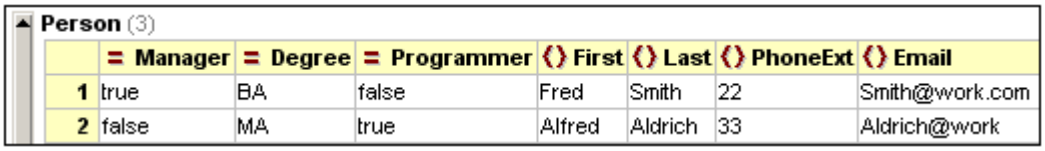
2. Wählen Sie die Menüoption **XML | Tabelle | Als Tabelle anzeigen** oder klicken Sie auf das Symbol . Beide `Person`-Elemente werden in Form einer einzigen Tabellen angezeigt. Die Element- und Attributnamen werden in der Kopfzeile der Tabelle, die einzelnen Elementinstanzen in den Tabellenzeilen angezeigt.



3. Wählen Sie die Menüoption **Ansicht | Optimale Breite** oder klicken Sie auf das Symbol , um die Spaltenbreite zu optimieren.

Hinweis: Sie können für einzelne Elementtypen die Tabellenansicht ein- oder ausblenden. Wählen Sie dazu die Tabelle aus (klicken Sie auf den Elementnamen in der Tabelle) und klicken Sie anschließend auf das Symbol "Als Tabelle anzeigen" . Beachten Sie jedoch, dass Child-Elemente, die als Tabellen angezeigt wurden, weiterhin als Tabellen angezeigt werden.

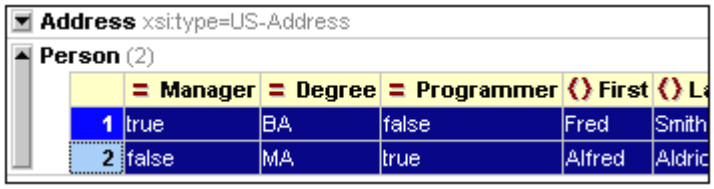
Eingabe von Inhalt in der Tabellenansicht
Um Inhalt für das zweite Person-Element einzugeben, doppelklicken Sie in jede der Tabellenzellen in der zweiten Zeile und geben Sie Daten ein. Beachten Sie jedoch, dass `PhoneExt` eine Ganzzahl bis 99 sein muss, damit die Datei gültig ist. Die intelligenten Editierfunktionen sind auch innerhalb der Tabellenzellen aktiv, sodass Sie Optionen aus Dropdown-Listen auswählen können, falls Optionen verfügbar sind (Boolscher Inhalt und die Enumerationen für das Attribut `Degree`).



Hinweis: Die Eingabehilfen sind auch für die als Tabelle dargestellten Elemente und Attribute aktiv. Wenn Sie z.B. in der Element-Eingabehilfe auf den Eintrag `Person` doppelklicken, wird eine neue Tabellenzeile hinzugefügt (D.h. es wird eine neue Instanz zum Element `Person` hinzugefügt).

Kopieren von XML Daten aus und in Drittprodukte
In XMLSpy 2005 können Kalkulationstabellendaten aus und in Produkte anderer Anbieter schnell und einfach kopiert werden. Die kopierten Daten können in XMLSpy 2005 sowie in Produkten anderer Anbieter in nativem Format der Anwendung verwendet werden. In diesem Abschnitt erfahren Sie, wie man Daten in und aus einem Excel-Datenblatt kopiert. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie auf die Zeilennummerierung 1, halten Sie die **Strg**-Taste gedrückt und klicken Sie auf die Zeilennummerierung 2. Hierdurch werden beide Zeilen markiert.



2. Wählen Sie die Menüoption **Bearbeiten | Als strukturierten Text kopieren**.

 © 2005 Altova GmbH

Der Befehl kopiert die Elemente in die Zwischenablage, und zwar genauso, wie sie am Bildschirm erscheinen.

3. Öffnen Sie Excel und fügen Sie die XML Daten (mit **Strg+V**) in ein Excel-Datenblatt ein.

A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	

4. Fügen Sie eine neue Zeile im Excel-Arbeitsblatt ein und geben Sie eine **dreistellige Zahl** im Element `PhoneExt` ein (z.B. `444`).

A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	
TRUE	Ph.D	FALSE	Colin	Coletti	444	Coletti@work.com	

5. Markieren Sie die Daten in Excel, wählen Sie **Bearbeiten | Kopieren**, um die Daten in die Zwischenablage zu kopieren
6. Wechseln Sie wieder zu XMLSpy 2005.
7. Klicken Sie in XMLSpy 2005 in die oberste linke Zelle der Tabelle und wählen Sie **Bearbeiten | Einfügen**.

Address xsi:type=US-Address

Person (3)

	= Manager	= Degree	= Programmer	{} First	{} Last	{} Pho
1	TRUE	BA	FALSE	Fred	Smith	22
2	FALSE	MA	TRUE	Alfred	Aldrich	33
3	TRUE	Ph.D	FALSE	Colin	Coletti	444

8. Die aktualisierten Daten wurden in die Tabelle übernommen.
7. Ändern Sie mittels der Menüoption **Suchen | Ersetzen** (Funktionstaste **Strg+H**) die Boolschen Werte `TRUE/FALSE` in `true/false` (in Kleinbuchstaben).

Sortieren der Tabellen nach Spalteninhalt

In der Tabellenansicht können Sie Ihre Daten je nach Spalte in aufsteigender oder absteigender Reihenfolge sortieren. Wir möchten nun die Tabelle `Person` alphabetisch nach Nachnamen sortieren.

1. Klicken Sie auf die Kopfzeile in der Spalte **Last**.

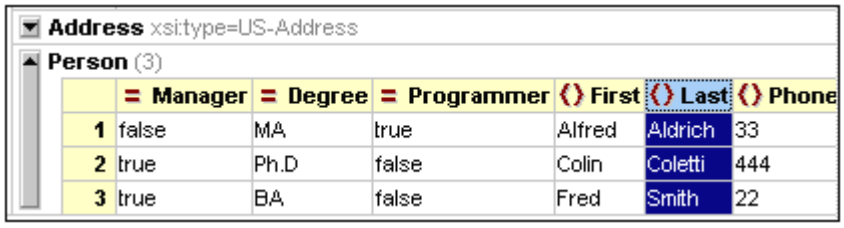
Address xsi:type=US-Address

Person (3)

	= Manager	= Degree	= Programmer	{} First	{} Last	{} Phone
1	true	BA	false	Fred	Smith	22
2	false	MA	true	Alfred	Aldrich	33
3	true	Ph.D	false	Colin	Coletti	444

2. Wählen Sie die Menüoption **XML | Tabelle | Aufsteigend sortieren** oder klicken Sie auf das Symbol .
Die Spalte wird unter Einbeziehung der gesamten Tabelle alphabetisch sortiert. Die

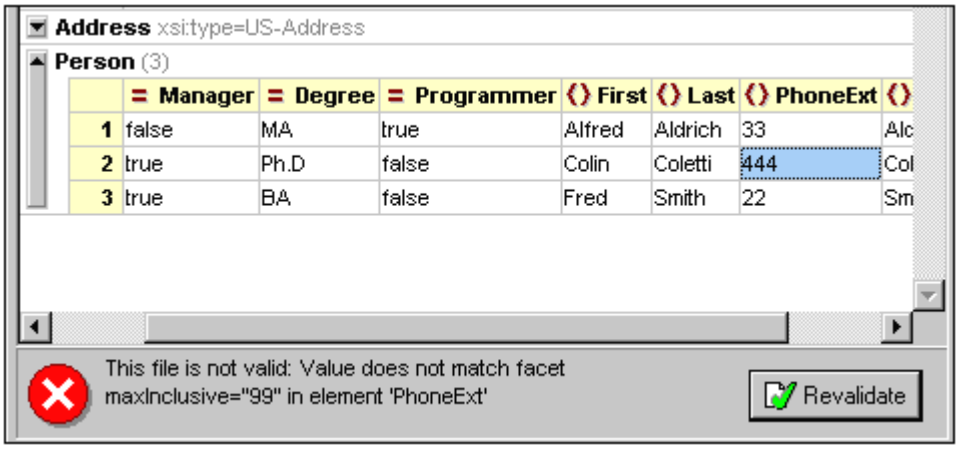
Spalte bleibt hierbei markiert.



	Manager	Degree	Programmer	First	Last	Phone
1	false	MA	true	Alfred	Aldrich	33
2	true	Ph.D	false	Colin	Coletti	444
3	true	BA	false	Fred	Smith	22

Die Tabelle wird nicht nur in der Ansicht sortiert, sondern die Daten werden auch im zugrundeliegenden XML-Dokument sortiert, d.h. die `Person`-Elemente werden nun alphabetisch nach dem Inhalt der Spalte `Last` sortiert. (Klicken Sie auf das Register "Text", um die Änderungen in der Textansicht zu sehen).

3. Wählen Sie die Menüoption **XML | Validieren** oder drücken Sie die Funktionstaste **F8**. Es wird eine Fehlermeldung angezeigt, dass der Inhalt eines `PhoneExt`-Elements nicht der `PhoneExt`-Facet `maxInclusive="99"` entspricht. Das fehlerverursachende PhoneExt-Element erscheint markiert.



	Manager	Degree	Programmer	First	Last	PhoneExt	
1	false	MA	true	Alfred	Aldrich	33	Alc
2	true	Ph.D	false	Colin	Coletti	444	Col
3	true	BA	false	Fred	Smith	22	Sm

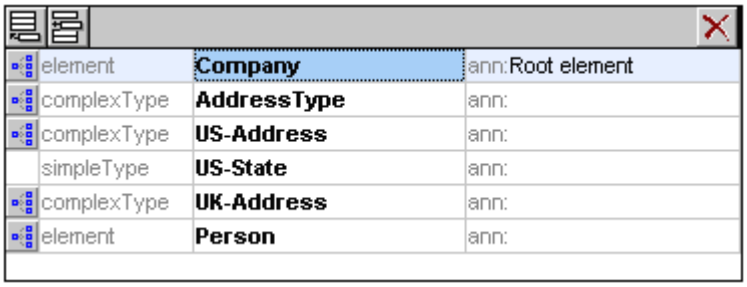
Da diese Durchwahlnummer sich außerhalb des von uns festgelegten Wertebereichs für `PhoneExt` befindet, müssen wir das XML-Schema ändern, damit die Nummer gültig ist. Dies wird im nächsten Abschnitt erläutert.

Ändern des Schemas

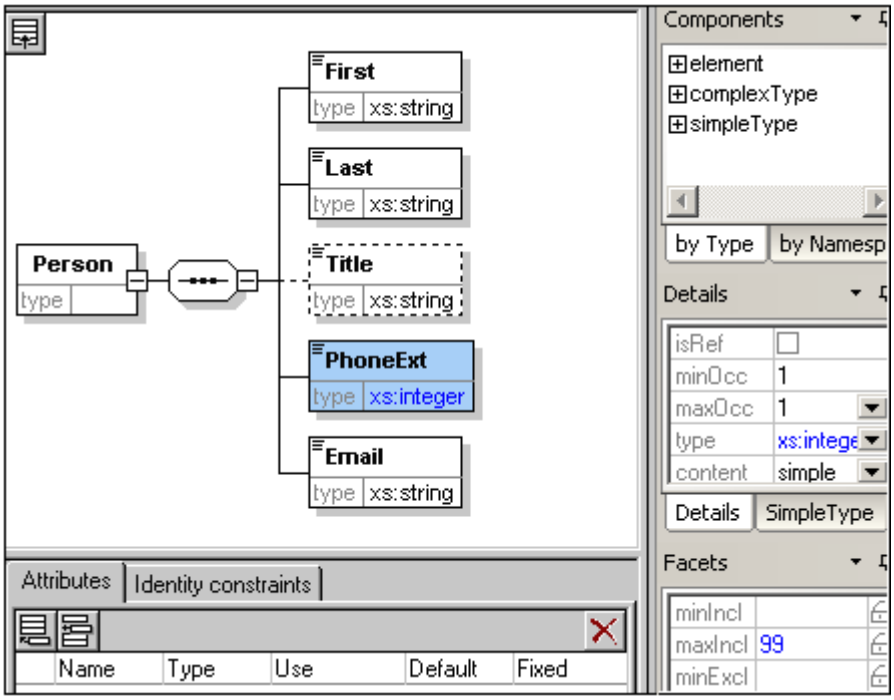
Wir haben nun gesehen, dass zweistellige Telefondurchwahlnummern nicht ausreichend sind. Wir werden daher den Wertebereich gültiger Nummern auf drei Stellen erweitern. Sie können das zugrunde liegende XML-Schema öffnen und ändern, ohne Ihr XML-Dokument schließen zu müssen.

1. Wählen Sie die Menüoption **DTD/Schema | Gehe zu Definition** oder klicken Sie auf das Symbol **.** Das assoziierte Schemadokument (in diesem Fall `AddressLast.xsd`) wird geöffnet. Wechseln Sie in die Schema/WSDL-Ansicht (siehe Abbildung unten).

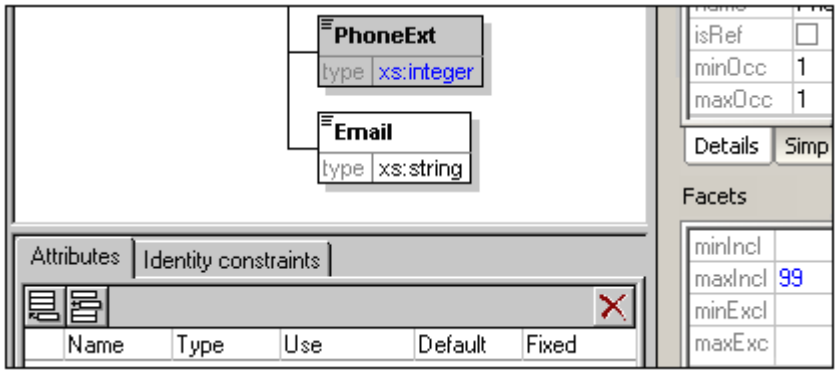
© 2005 Altova GmbH



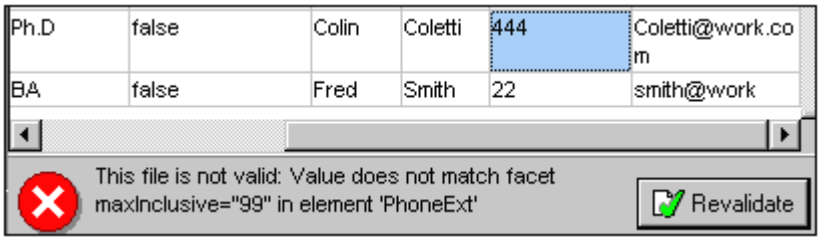
2. Klicken Sie auf das Symbol "Diagramm anzeigen" des globalen Elementes `Person` und klicken anschließend auf das Element `PhoneExt`. Die Facet-Daten werden auf dem Register "Facets" angezeigt.



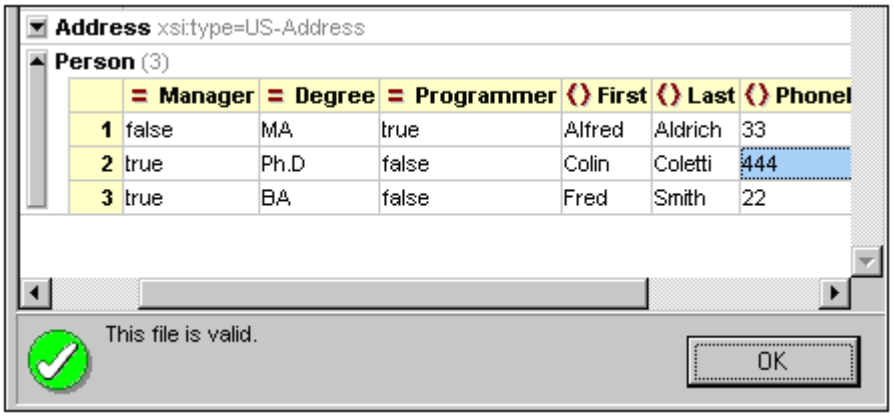
3. Doppelklicken Sie auf dem Register "Facets" auf das Wertefeld `maxIncl`, geben Sie `999` als Maximalwert ein und drücken Sie die Eingabetaste.



4. Speichern Sie das Schema-Dokument.
5. Mit **Strg+TAB** kommen Sie zurück zum XML-Dokument.



6. Klicken Sie auf die Schaltfläche **Revalidate,** um das XML-Dokument erneut zu validieren.



Die Meldung "`Diese Datei ist gültig`" wird angezeigt. Das XML-Dokument ist nun mit der geänderten Schemadefinition konform.

6. Wählen Sie die Menüoption **Datei | Speichern unter...** und benennen Sie die Datei `CompanyLast.xml`. (Denken Sie daran, das Originaldokument `CompanyLast.xml` umzubenennen z.B. in `CompanyLast_orig.xml`).

Hinweis: Das Originaldokument `CompanyLast.xml` ist im Verzeichnis `Tutorial` abgelegt.

2.1.6 Transformieren von XML mit XSLT

Ihr Ziel in diesem Abschnitt
Erstellen einer HTML-Datei von einer XML-Datei mittels eines XSLT Stylesheet zum Transformieren der XML-Datei. Beachten Sie, dass die XML-Datei bei einer "Transformation" nicht geändert wird. Es wird nur eine neue Ausgabedatei erzeugt. Das Wort "Transformation" ist eine Konvention.

Methode:
Gehen Sie beim Transformieren folgendermaßen vor:

- Weisen Sie dem XML-Dokument eine vordefinierte XSL-Datei, nämlich `company.xsl` zu.
- Führen Sie die Transformation mit Hilfe eines der beiden in XMLSpy 2005 integrierten Altova XSLT-Prozessoren durch (*siehe Hinweis unten*).

Die folgenden XMLSpy 2005-Befehle werden in diesem Abschnitt verwendet:

XSL/XQuery | XSL zuweisen weist dem aktiven XML-Dokument eine XSL-Datei zu.

 © 2005 Altova GmbH

XSL/XQuery | Gehe zu XSL öffnet die vom aktiven XML-Dokument referenzierte XSL-Datei.

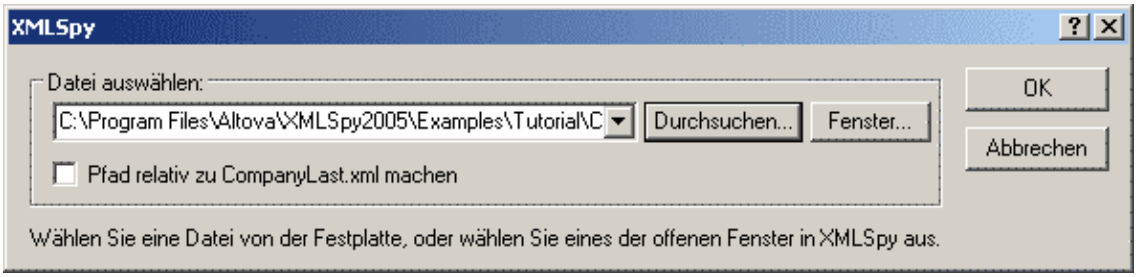
XSL/XQuery | XSL-Transformation (F10) oder die Symbolleistenschaltfläche transformiert das aktive XML-Dokument mit Hilfe des diesem zugewiesenen XSL Stylesheet. Wenn keine XSL-Datei zugewiesen wurde, werden Sie beim Aufrufen dieses Befehls aufgefordert, eine zuzuweisen.

Hinweis: XMLSpy 2005 verfügt über zwei integrierte XSLT-Prozessoren, den Altova XSLT 1.0-Prozessor und den Altova XSLT 2.0-Prozessor. Der Altova XSLT 1.0-Prozessor dient zum Transformieren von XSLT 1.0 Stylesheets, während der Altova XSLT 2.0-Prozessor für die Verarbeitung von XSLT 2.0 Stylesheets verwendet wird. Der richtige Prozessor wird von XMLSpy 2005 automatisch auf Basis des Versionsattributs im Element `xsl:stylesheet` oder `xsl:transform` ausgewählt. In diesem Tutorial verwenden wir für die Transformation XSLT 1.0 Stylesheets. Bei Aufruf des Befehls **XSL-Transformation** wird bei diesen Stylesheets automatisch der Altova XSLT 1.0-Prozessor verwendet.

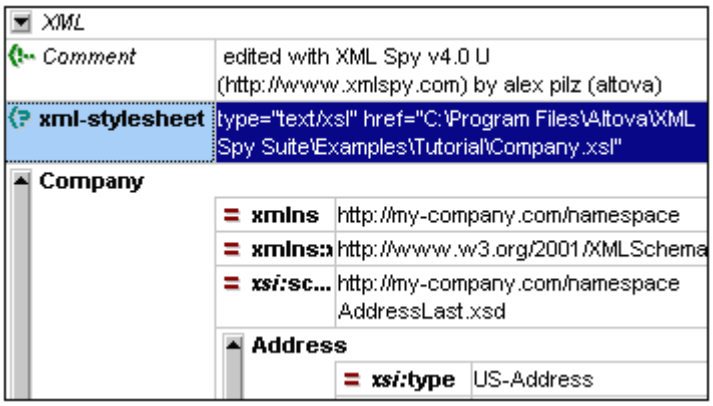
Zuweisen einer XSL-Datei

So weisen Sie der Datei `CompanyLast.xml` eine XSL-Datei zu:

1. Klicken Sie im Hauptfenster auf das Register `CompanyLast.xml`, um es aktiv zu setzen.
2. Wählen Sie die Menüoption **XSL/XQuery | XSL zuweisen**.
3. Klicken Sie auf die Durchsuchen-Schaltfläche und wählen Sie im Ordner `Tutorial` die Datei `Company.xsl` aus. Sie können im Dialogfeld die Option "Pfad relativ zu CompanyLast.xml machen", aktivieren, um den Pfad zur XSL-Datei (im XML-Dokument) relativ zu machen.



4. Klicken Sie auf OK um dem XML-Dokument die XSL-Datei zuzuweisen.



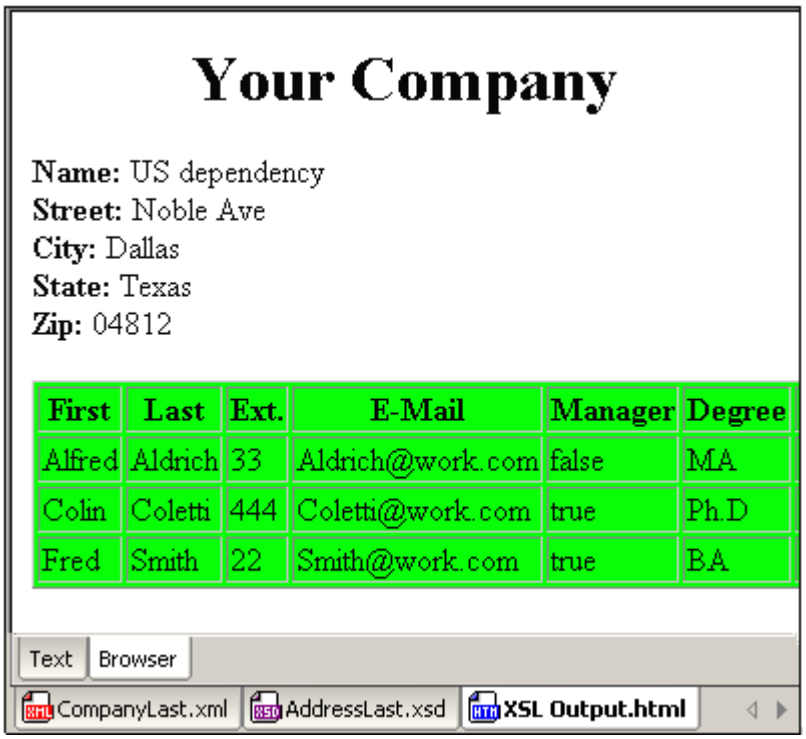
Im XML-Dokument wird eine `XML-stylesheet` Verarbeitungsanweisung eingefügt, die

die XSL-Datei referenziert. Wenn Sie das Kontrollkästchen "Pfad relativ zu CompanyLast.mxl machen" aktiviert haben, ist der Pfad relativ; andernfalls ist er absolut (wie in der obigen Abbildung).

Transformieren der XSL-Datei

Um das XML-Dokument mit Hilfe der zugewiesenen XSL-Datei zu transformieren, gehen Sie folgendermaßen vor:

1. Stellen Sie sicher, dass die XML-Datei das aktive Dokument ist.
2. Wählen Sie die Menüoption **XSL/XQuery | XSL-Transformation (F10)** oder klicken Sie auf das Symbol , um die Transformation mittels des in dem XML-Dokument referenzierten XSL Stylesheet zu starten. (Da die Datei `Company.xsl` ein XSLT 1.0-Dokument ist, wird für die Transformation automatisch der integrierte Altova XSLT 1.0-Prozessor ausgewählt). Das Ausgabedokument wird in der Browser-Ansicht angezeigt. Es hat den Namen `XSL Output.html`. Die Firmendaten werden in einem Block links oben angezeigt, während die Personendaten unterhalb davon in Tabellenform dargestellt werden.



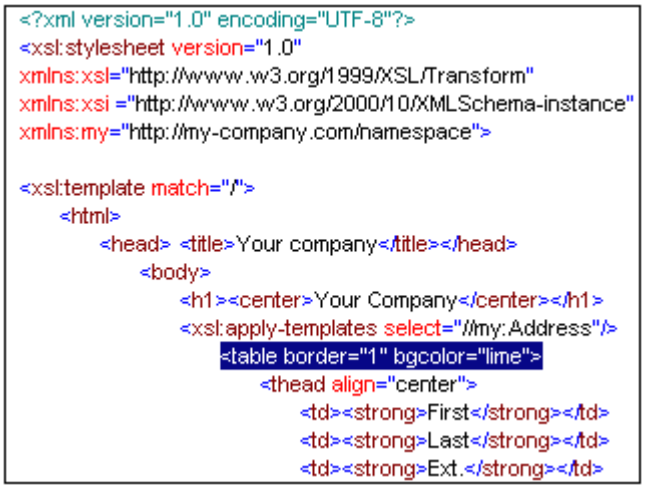
Hinweis: Falls in der Ausgabedatei nur eine Tabellenkopfzeile aber keine Tabellendaten zu sehen sind, stellen Sie bitte sicher, dass Sie den Target Namespace für Ihr Schema wie am Beginn des Tutorials unter Anlegen einer neuen XML-Schema-Datei beschrieben, definiert haben. Der Namespace muss in allen drei Dateien (Schema, XML und XSL) **identisch** sein.

© 2005 Altova GmbH

Ändern der XSL-Datei

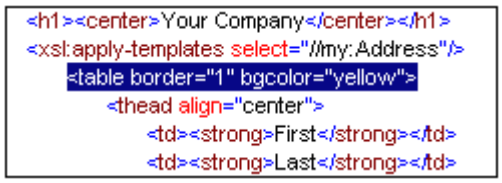
Sie können die Ausgabe der Datei ändern, indem Sie die zugrunde liegende XSL-Datei bearbeiten. In diesem Fall soll die Hintergrundfarbe der Tabelle von "Lime (Hellgrün)" in "Yellow (Gelb)" geändert werden.

1. Klicken Sie auf das Register `CompanyLast.xml` um es aktiv zu setzen und stellen Sie sicher, dass Sie sich in der Grid-Ansicht befinden.
2. Wählen Sie die Menüoption **XSL/XQuery | Gehe zu XSL**.

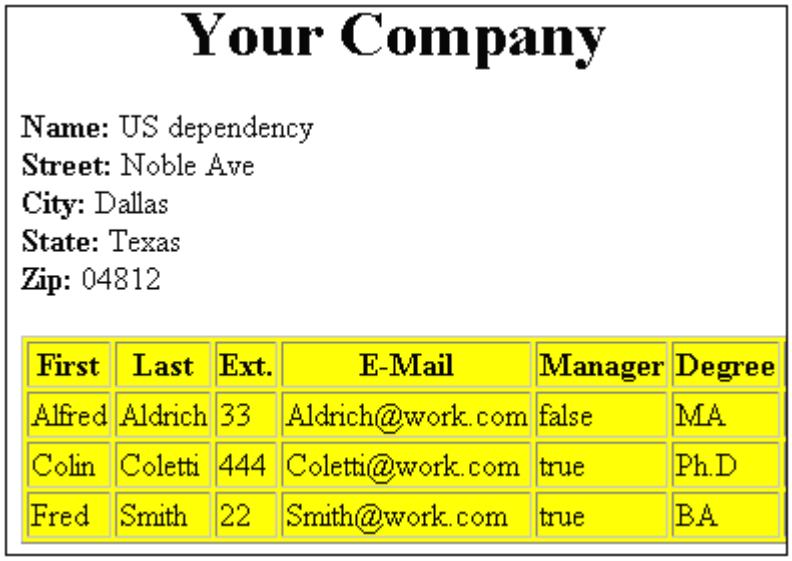


Mit diesem Befehl wird die Datei `Company.XSL` geöffnet, die in der XML-Datei referenziert wird.

3. Gehen Sie zur Zeile `<table border="1" bgcolor="lime">` und ändern Sie den Eintrag von `bgcolor="lime"` in `bgcolor="yellow"`.



4. Wählen Sie die Menüoption **Datei | Speichern,** um die Änderungen an der XSL-Datei zu speichern.
5. Klicken Sie auf das Register `CompanyLast.xml` um die XML-Datei aktiv zu setzen, und wählen Sie **XSL/XQuery | XSL-Transformation** oder drücken Sie die Funktionstaste **F10**. In der Browser-Ansicht von XMLSpy 2005 erscheint eine neue Datei namens `XSL Output.html`. Die Hintergrundfarbe der Tabelle ist gelb.



Your Company

Name: US dependency
Street: Noble Ave
City: Dallas
State: Texas
Zip: 04812

First	Last	Ext.	E-Mail	Manager	Degree
Alfred	Aldrich	33	Aldrich@work.com	false	MA
Colin	Coletti	444	Coletti@work.com	true	Ph.D
Fred	Smith	22	Smith@work.com	true	BA

6. Wählen Sie die Menüoption **Datei | Speichern** und benennen Sie die Datei `Company.html`.

2.1.7 Arbeiten mit Datenbanken

Ihr Ziel in diesem Abschnitt:
Exportieren Sie Informationen zu einer bestimmten Person aus der Adressenliste nach MS Access, und re-importieren Sie die Daten nach XMLSpy 2005.

Dies erfolgt durch:

- Verwendung der Menüoption **Konvertieren** und Auswahl des Import - oder Exportprozesses

Funktionen in diesem Abschnitt:

Konvertieren | **Exportieren in Textdatei/Datenbank**; mit dieser Funktion können Sie XML-Daten als Text oder zur Verwendung in Datenbanken von Fremdanbietern exportieren.

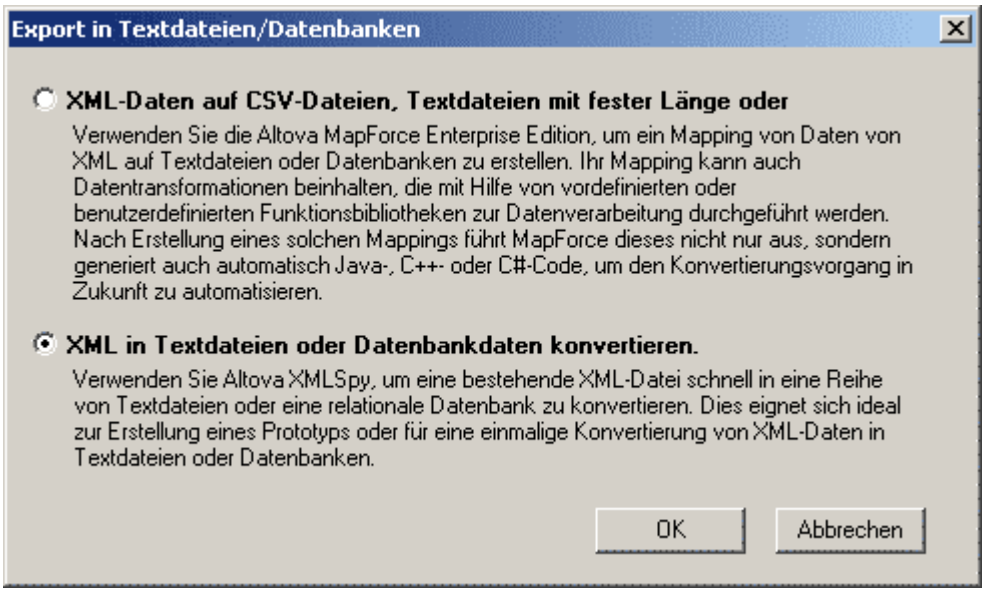
Konvertieren | **Datenbank importieren**; mit dieser Funktion können Sie Daten aus Datenbanken nach XMLSpy 2005 importieren.

Exportieren von XML-Daten in externe Datenbanken

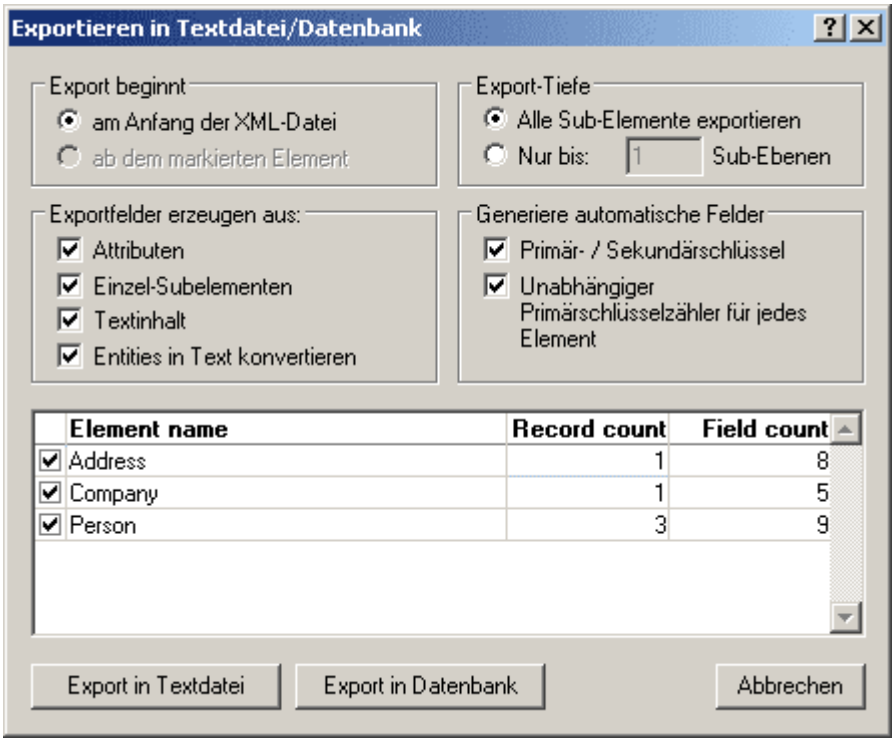
So exportieren Sie Daten in eine Datenbank:

1. Klicken Sie im Hauptfenster auf das Register **CompanyLast.xml,** um das Dokument aktiv zu setzen.
2. Wählen Sie in der Menüleiste **Konvertieren | Exportieren in Textdatei / Datenbank...**.

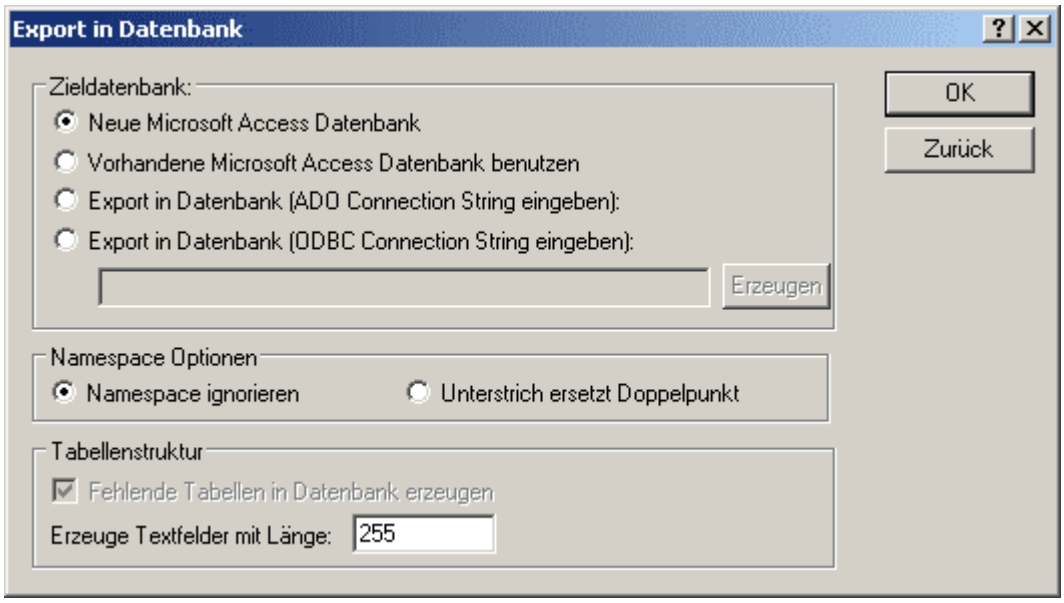
© 2005 Altova GmbH



3. Klicken Sie auf das Optionsfeld "**XML in Textdateien oder Datenbankdaten konvertieren**" und bestätigen Sie mit OK. Mit den vordefinierten Einstellungen können Sie in diesem Dialog alle Elemente und Attribute exportieren, sowie Primär- und Sekundärschlüssel erstellen.

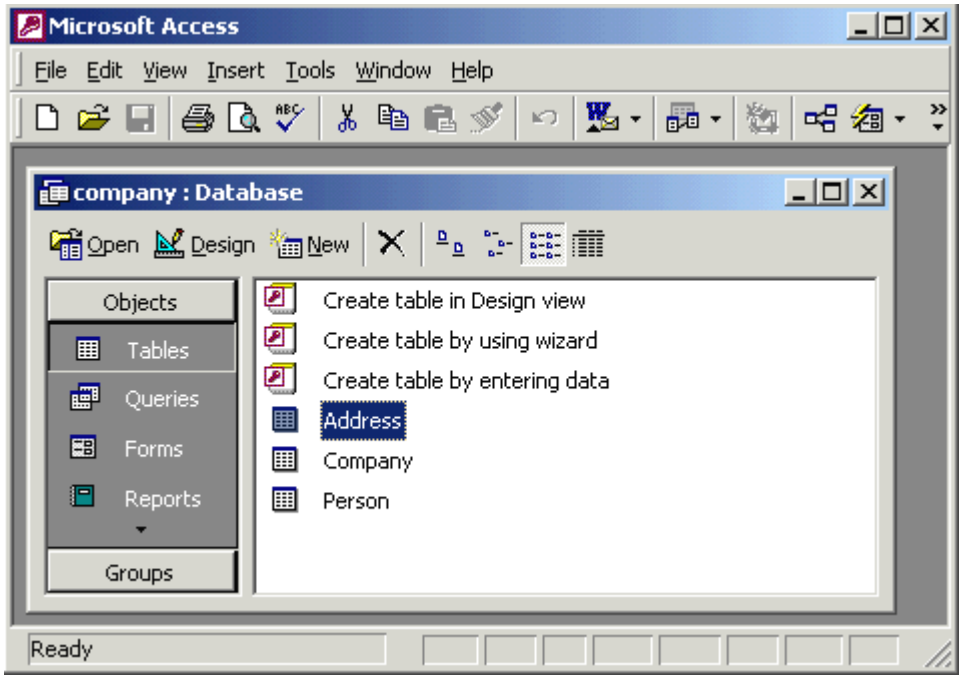


4. Klicken Sie auf die Schaltfläche **Export in Datenbank**.



In diesem Dialog können Sie auswählen, ob Sie eine neue Access-Datenbank erstellen möchten, Informationen in eine bereits vorhandene Datenbank oder in eine Datenbank eines Drittanbieters exportieren möchten. Die Option "Namespace ignorieren" ist standardmäßig aktiv.

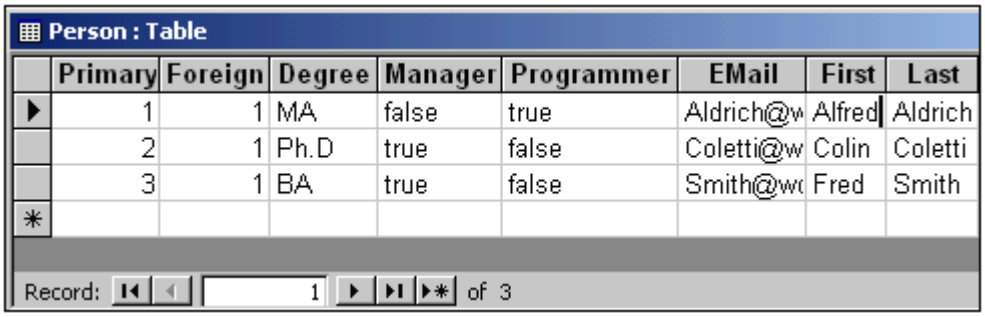
5. Klicken Sie auf den Eintrag **Neue Microsoft Access Datenbank** und bestätigen Sie mit OK.
6. Geben Sie den Namen für die neue Datenbank im Dialog "Speichern unter..." (z.B. Company.mdb) ein und bestätigen Sie durch Klicken auf die Schaltfläche "Speichern". Der Exportfortschritt wird durch einen Balken angezeigt. Sie erhalten dann eine Nachricht über den erfolgten Export. Bestätigen Sie mit OK.
7. Öffnen Sie die Datei **Company.mdb**, die Sie zuvor gespeichert hatten. Durch den Export-Prozess wurde automatisch eine Tabelle für jedes exportierte Element erstellt.



8. Doppelklicken Sie auf das Symbol **Person** um die Personentabelle zu öffnen.

© 2005 Altova GmbH

Die Tabelle zeigt alle Informationen an, die in der XML-Datei verfügbar sind. Des Weiteren wurden die "Automatischen Felder" Primär- und Sekundärschlüssel erstellt, mit denen die Datenbank indiziert werden kann.



Bitte beachten Sie:

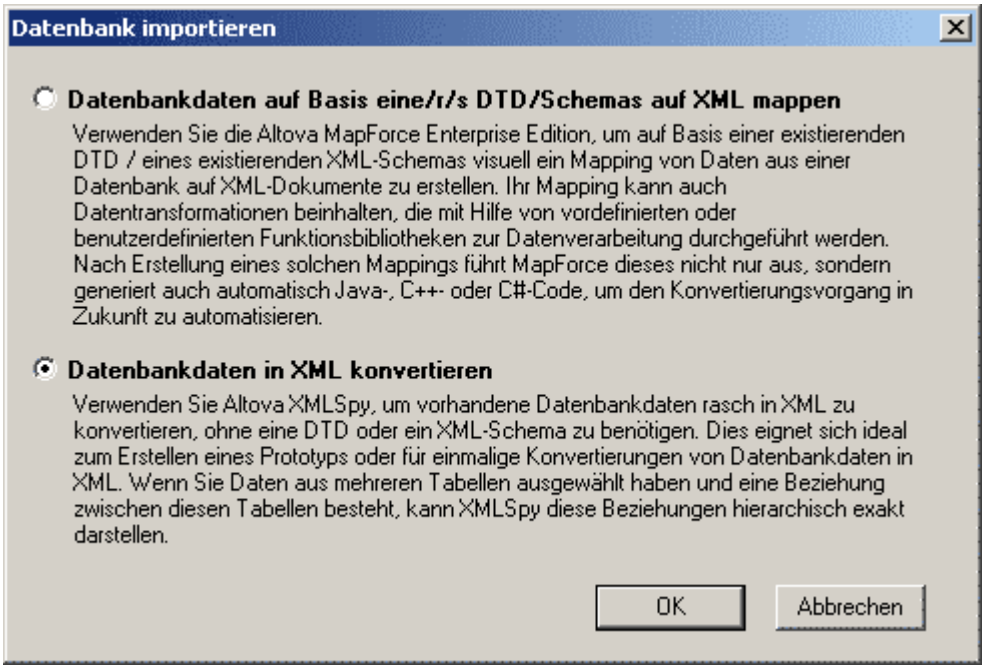
Beim Auswählen der Option **Neue Microsoft Access Datenbank** beim Exportieren von Datenbankeninformationen, erstellt XMLSpy 2005 eine neue Access 2000 Datenbank!

Sollten Sie Daten in eine Access 97 Datenbank exportieren wollen, müssen Sie zuerst eine neue leere Access 97 Datenbank erstellen. Danach wählen Sie die Option **Vorhandene Microsoft Access Datenbank benutzen**. Für den Import von Daten aus Access Datenbanken bestehen hingegen keine Einschränkungen.

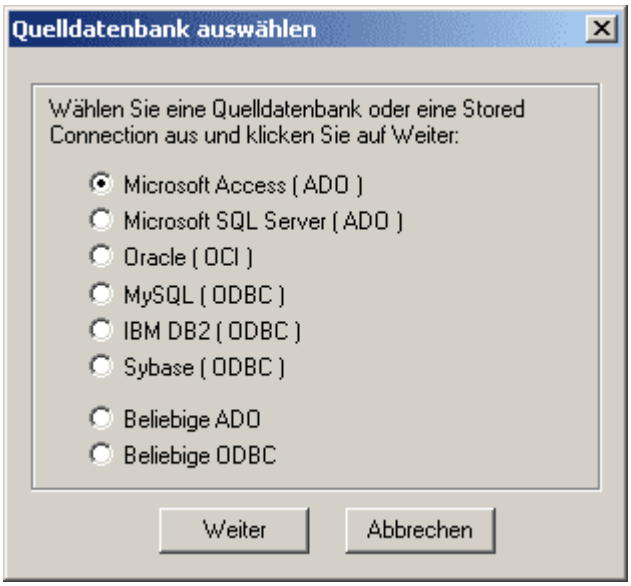
Importieren von Datenbankeninformationen

So importieren Sie Daten in XMLSpy 2005:

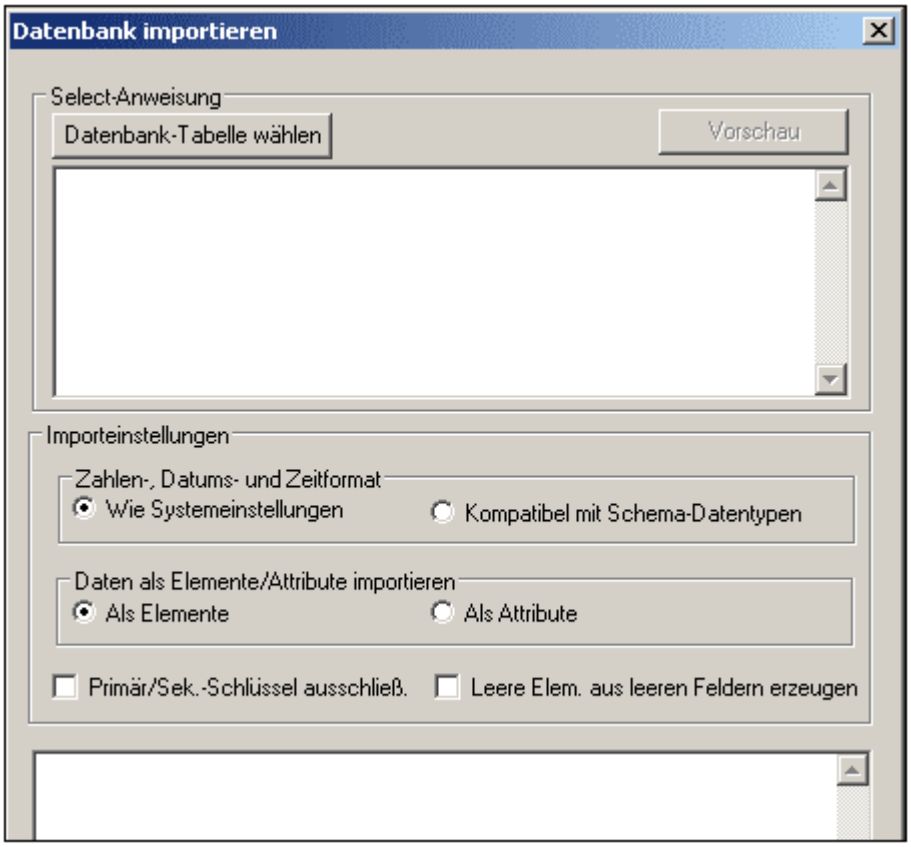
1. Wählen Sie die Menüoption **Konvertieren | Datenbank importieren...**. Daraufhin öffnet sich das Dialogfeld "Datenbank importieren".



2. Klicken Sie auf die Schaltfläche **Datenbankdaten in XML konvertieren** und bestätigen Sie mit OK. Daraufhin wird das Dialogfeld "Quelldatenbank auswählen" angezeigt.



3. Wählen Sie Microsoft Acces (ADO) und klicken Sie auf **Weiter**. Das Dialogfeld "MS Access-Datenbank auswählen" wird geöffnet.
4. Klicken Sie in diesem Dialogfeld auf die Schaltfläche **Durchsuchen** und suchen Sie die Datei **Company.mdb**. Klicken Sie auf **Weiter**, um das Dialogfeld zum Import von Datenbankdaten zu öffnen.

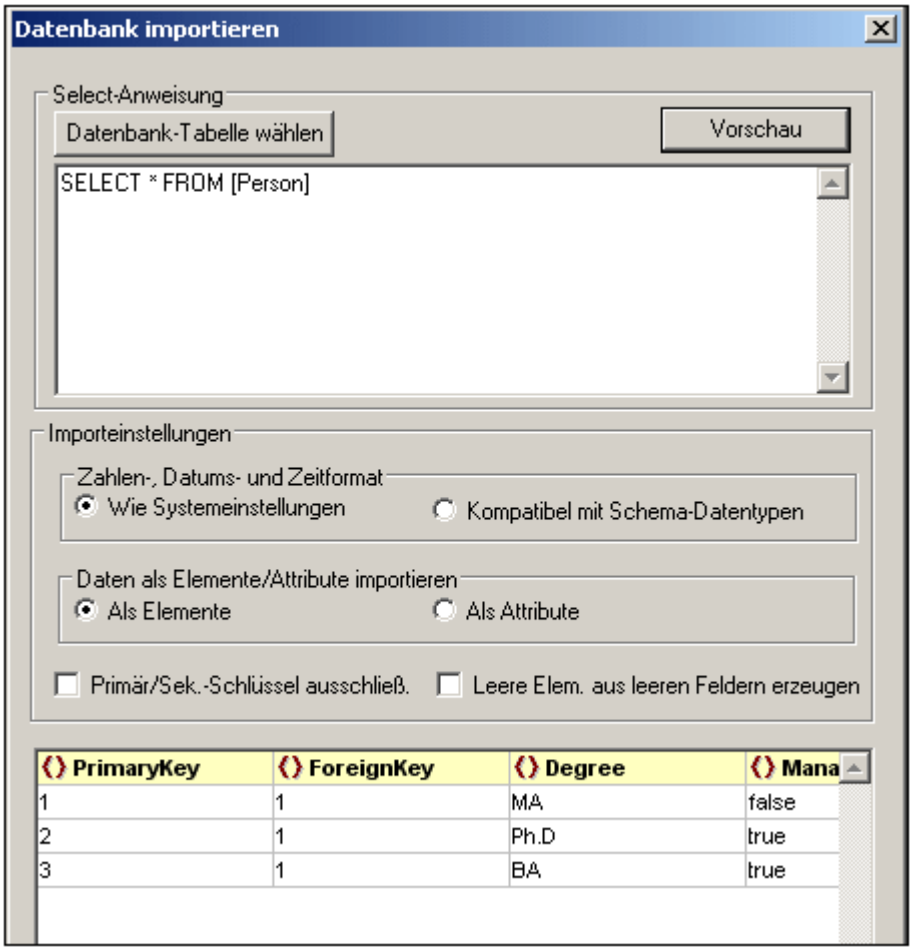


5. Klicken Sie auf die Schaltfläche **Datenbank-Tabelle wählen**, wählen Sie Person und

© 2005 Altova GmbH

bestätigen Sie mit OK.
Daraufhin wird ins Textfeld die `SELECT * from [Person]` Anweisung eingefügt.

6. Klicken Sie auf **Vorschau** um eine Voransicht der zu importierenden Daten zu erhalten. Im Vorschau-Fenster werden nur die Tabellendaten angezeigt, die mit den ausgewählten Kriterien übereinstimmen.



Felder importieren: als Attribute, Elemente oder Import überspringen
Standardmäßig werden alle Felder als Elemente importiert. Mit Hilfe des Vorschau-Fensters können Sie für die einzelnen Felder den XML Node-Typ (Element oder Attribut) angeben und festlegen, welche Felder importiert werden sollen. Durch mehrfaches **Anklicken des Element-Symbols** {} links neben dem Element-Namen können Sie die verschiedenen verfügbaren Optionen auswählen:

{} Dieses Feld als **Element** importieren.

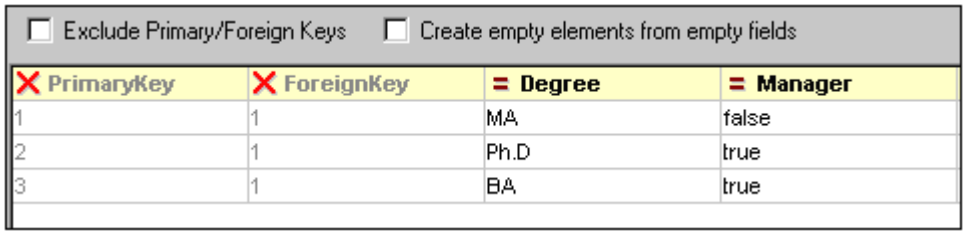
Dieses Feld als **Attribut** importieren.

Überspringen, dieses Feld nicht importieren.

Geben Sie nun für jedes Feld den XML Node-Typ an oder legen Sie fest, ob das Feld importiert werden soll. Gehen Sie dabei folgendermaßen vor:

1. Klicken Sie auf das Symbol **Element** in der Spalte PrimaryKey, bis das Symbol **Überspringen** erscheint. Wiederholen Sie diese Aktion in der Spalte ForeignKey.

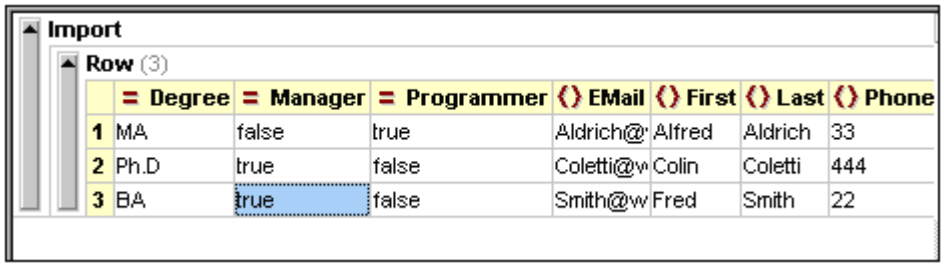
2. Klicken Sie auf das Symbol **Element** in der Spalte "Degree", bis das Symbol **Attribut** erscheint. Wiederholen Sie diese Aktion in den Spalten Manager und Programmer.



Exclude Primary/Foreign Keys Create empty elements from empty fields

PrimaryKey	ForeignKey	Degree	Manager
1	1	MA	false
2	1	Ph.D	true
3	1	BA	true

3. Klicken Sie auf die Schaltfläche **Import**, um den Importvorgang zu starten. XMLSpy 2005 erstellt hierdurch eine **unbenannte XML-Datei**, welche die Personen-Tabellendaten enthält. Das **Root-Element** ist nun Import, jedes Person Element ist jetzt ein Row-Element.



Import

Row (3)

	Degree	Manager	Programmer	EMail	First	Last	Phone
1	MA	false	true	Aldrich@	Alfred	Aldrich	33
2	Ph.D	true	false	Coletti@w	Colin	Coletti	444
3	BA	true	false	Smith@w	Fred	Smith	22

Klicken Sie auf das Register Text, um eine andere Ansicht der importierten Daten zu erhalten.

```
<Import>
    <Row Degree="MA" Manager="false" Programmer="true">
        <EMail>Aldrich@work.com</EMail>
        <First>Alfred</First>
        <Last>Aldrich</Last>
        <PhoneExt>33</PhoneExt>
    </Row>
    <Row Degree="Ph.D" Manager="true" Programmer="false">
        <EMail>Coletti@work.com</EMail>
        <First>Colin</First>
        <Last>Coletti</Last>
        <PhoneExt>444</PhoneExt>
    </Row>
    <Row Degree="BA" Manager="true" Programmer="false">
        <EMail>Smith@work.com</EMail>
        <First>Fred</First>
        <Last>Smith</Last>
        <PhoneExt>22</PhoneExt>
    </Row>
</Import>
```

Weitere Informationen zum Thema Datenimport finden Sie im Referenzhandbuch im Abschnitt DB-Daten importieren unter **Menüs | Menü "Konvertieren"**.

Erstellen eines Datenbank-Schemas

Mit XMLSpy 2005 können Sie Schemas erstellen, die auf externen Datenbankdateien basieren. Es werden die meisten gebräuchlichen Datenbanken sowie ADO- oder ODBC-kompatible Datenbanken unterstützt.

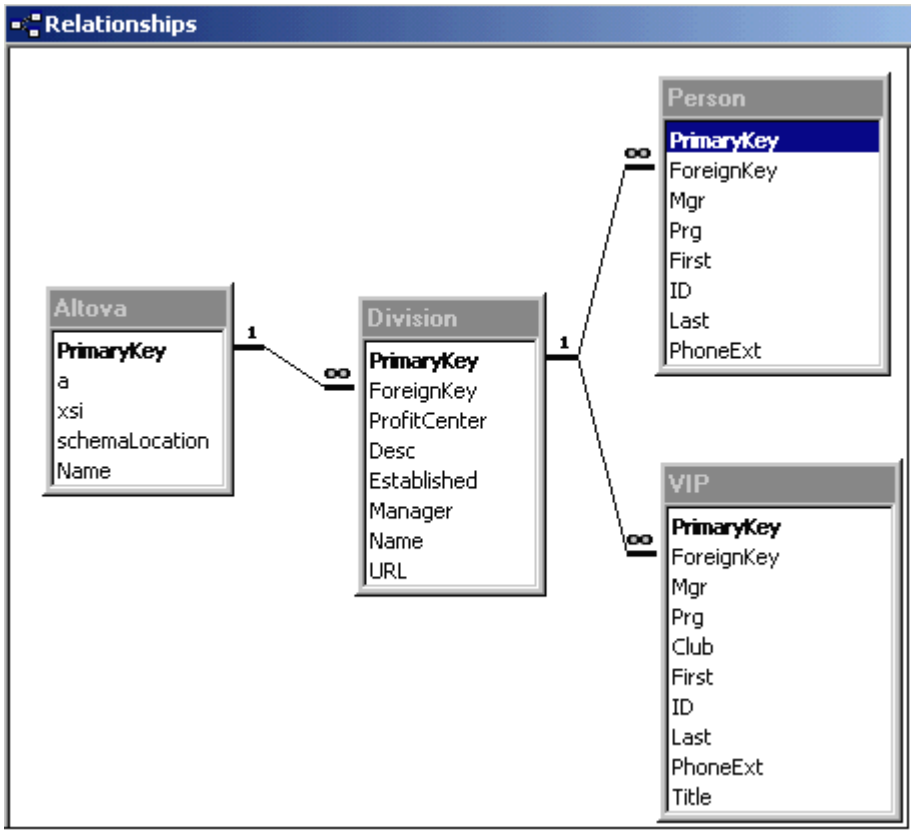
© 2005 Altova GmbH

Ziel dieses Abschnittes:
Konvertieren von MS Access Datenbanken in eine Schemadatei mit derselben Tabellenstruktur.

Dies wird erreicht durch:

- Verwendung der Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen** zum Erstellen eines Schemas in XMLSpy 2005.

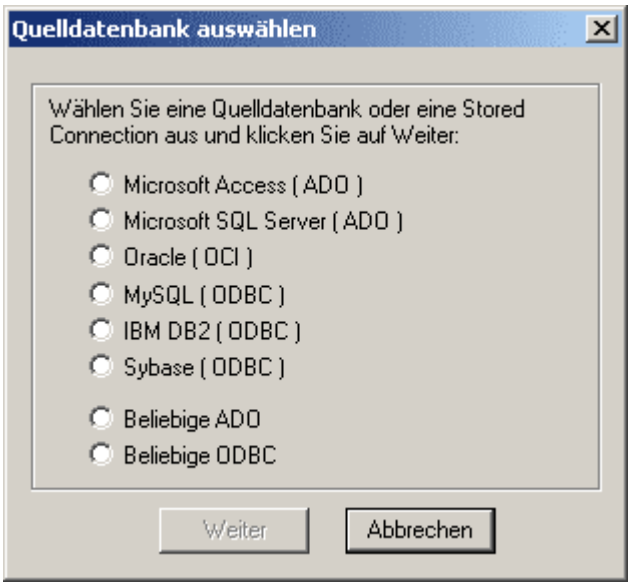
- In diesem Beispiel verwenden wir die Datei **DB2Schema.mdb**, die mit dieser Anleitung geliefert wird. Die Relationships-Ansicht der Datei DB2schema.mdb sehen Sie im Bild unten. Verwenden Sie die Menüoption **Tools | Relationships** in MS Access um die Datenhierarchiebeziehungen anzusehen.



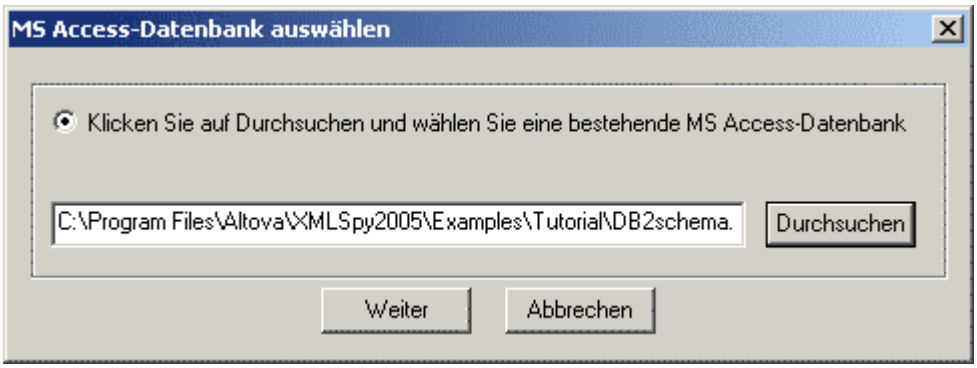
Konvertieren einer Datenbank in ein Schema

So erstellen Sie ein Schema anhand einer Datenbanken-Datei:

1. Wählen Sie die Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen.** Daraufhin wird das Dialogfeld **Quelldatenbank auswählen** angezeigt.

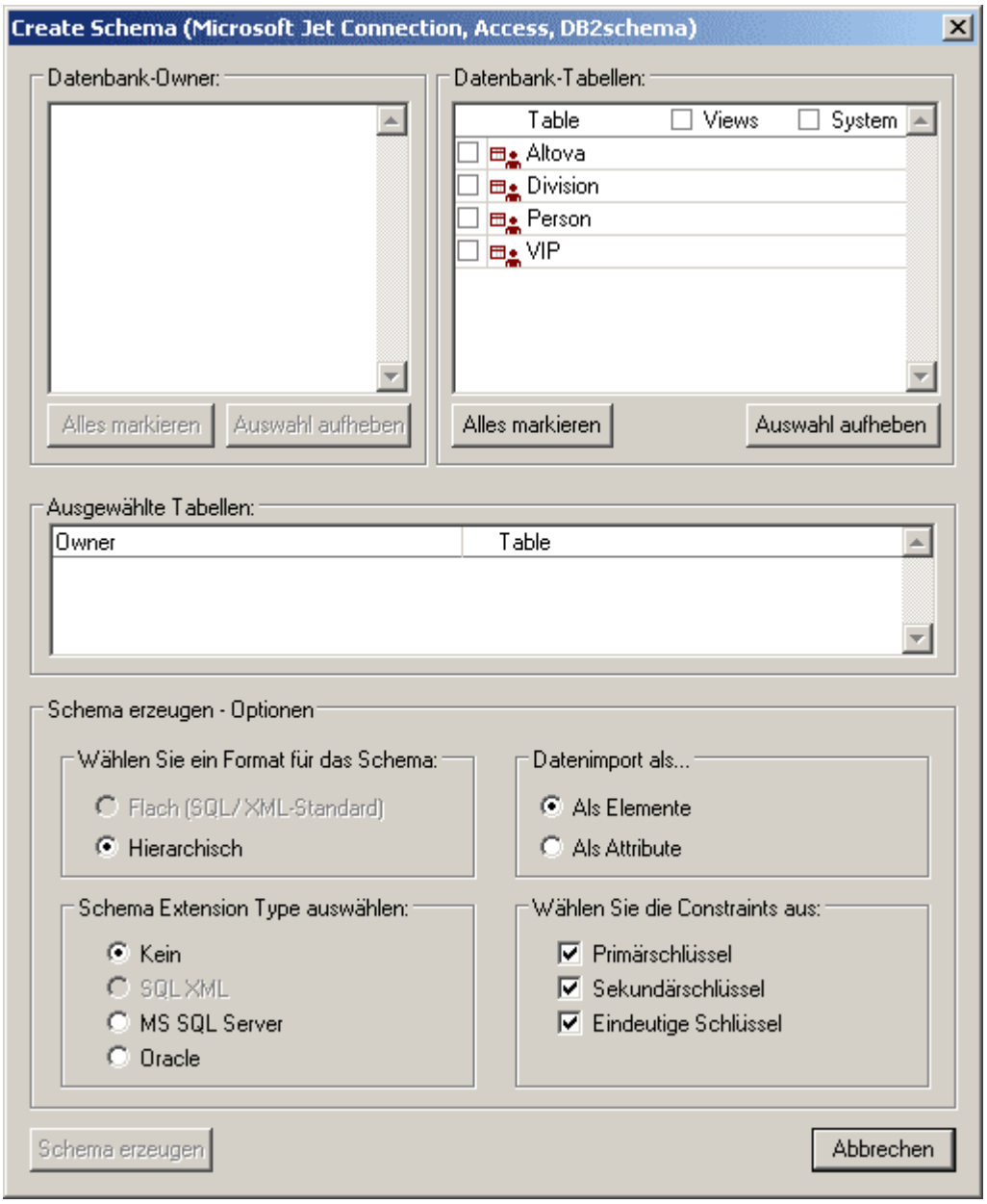


2. Wählen Sie **Microsoft Access (ADO)** und klicken Sie auf **Weiter**.
3. Klicken Sie auf die Durchsuchen-Schaltfläche und wählen Sie die Datei **DB2schema.mdb** im Verzeichnis `Examples\Tutorial` aus, so dass der Dateiname **DB2schema.mdb** im Dialogfeld aufscheint.

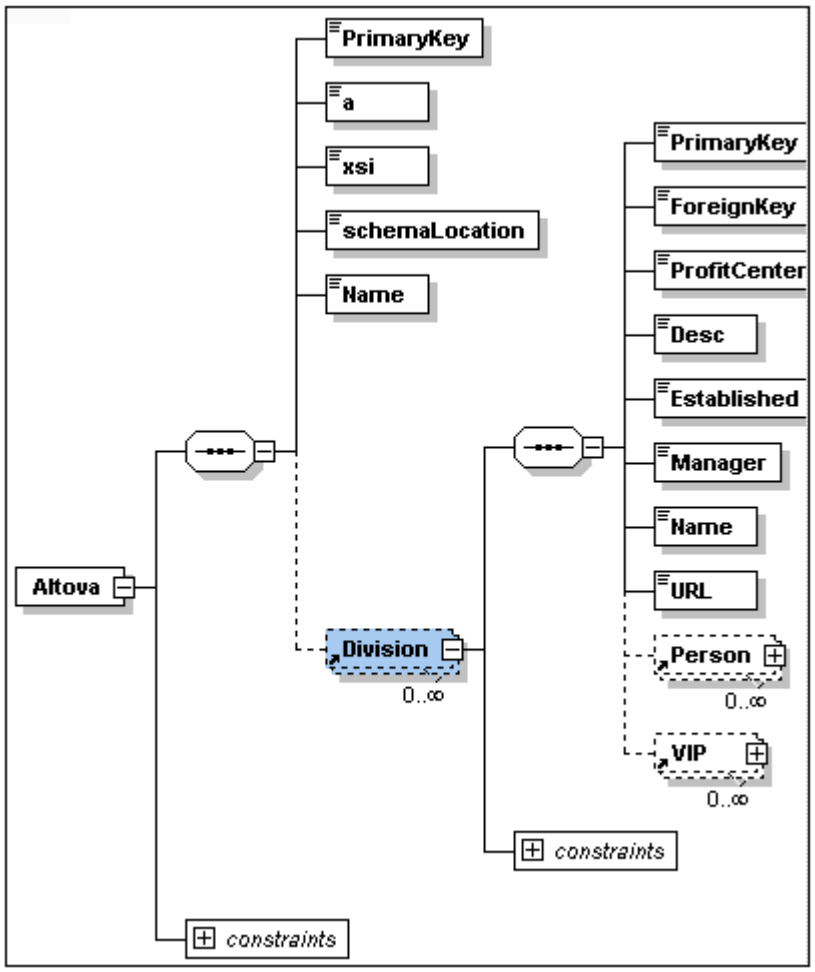


4. Klicken Sie auf **Weiter**. Es wird das Dialogfeld **Schema erzeugen** geöffnet, in dem Sie spezifische Datenbanktabellen und andere Schemaeinstellungen festlegen können.

© 2005 Altova GmbH



5. Klicken Sie im Fenster **Datenbank-Tabellen** auf die Schaltfläche **Alles markieren**, um alle Datenbanktabellen auszuwählen. Die ausgewählten Tabellen werden im Fenster **Ausgewählte Tabellen** angezeigt.
6. Durch Klicken auf die Schaltfläche **Schema erzeugen** starten Sie die Konvertierung. Das generierte Schema wird in der Schema/WSDL Design-Ansicht angezeigt.
7. Um die keyref- und key-Felder der einzelnen Elemente anzuzeigen, klicken Sie auf das Register "Identity constraints".
8. Klicken Sie auf das Komponentensymbol neben dem globalen Element **Altova**, um das Content Model anzuzeigen.



9. Wählen Sie die Menüoption **Datei | Speichern unter** und speichern Sie das neue Schema unter einem Namen, z.B. **DB2schema.xsd**.
10. Klicken Sie auf das Symbol **Globale Elemente anzeigen** , um zur Schema-Übersicht zurückzukehren.

Datenbanken, die key und keyref-Felder unterstützen:
MS Access sowie eine Vielzahl anderer Datenbanken sind in der Lage, automatisch key und keyref-Informationen an den ADO-Driver zu senden. Diese werden benötigt, um Datenbankenhierarchien darzustellen.

Andere Datenbanken: Konvertieren einer DB-Struktur in ein XML-Schema
Bei allen anderen Datenbanken mit Ausnahme von MS Access-Datenbanken erfolgt die Konvertierung in zwei Schritten:

- Erstellung einer Verbindungszeichenfolge für die Verbindung mit der DB
- Auswahl der zu konvertierenden DB-Tabellen

Nähere Informationen zum Konvertieren dieser Datenbankformate finden Sie im Benutzerhandbuch unter Erstellen eines XML-Schemas anhand einer DB-Struktur, Abschnitt (**Menüs | Konvertieren**).

© 2005 Altova GmbH

2.1.8 Projektverwaltung

In diesem Abschnitt werden Sie mit den Projektverwaltungsfunktionen von XMLSpy 2005 vertraut gemacht. Nachdem Sie die Vorteile der Gliederung von XML-Dateien in Projekten kennengelernt haben, werden wir die soeben erstellten Dateien in einem einfachen Projekt strukturieren.

Vorteile von Projekten

Im Folgenden sind die Vorteile der Strukturierung von XML-Dateien in Form von Projekten aufgelistet.

- Dateien und URLs können durch gemeinsame Erweiterungen oder andere Kriterien in Ordner gruppiert werden.
- Batchprozesse können bestimmten Ordnern oder dem ganzen Projekt zugewiesen werden.
- DTD / XML-Schemas können bestimmten Ordnern zugewiesen werden, um eine sofortige Validierung der darin befindlichen Dateien zu ermöglichen.
- XSL-Dateien können bestimmten Ordnern zugewiesen werden, um eine sofortige Transformation der darin befindlichen Dateien zu erlauben.
- Es können Zielordner für die XSL-Transformationsdateien definiert werden.

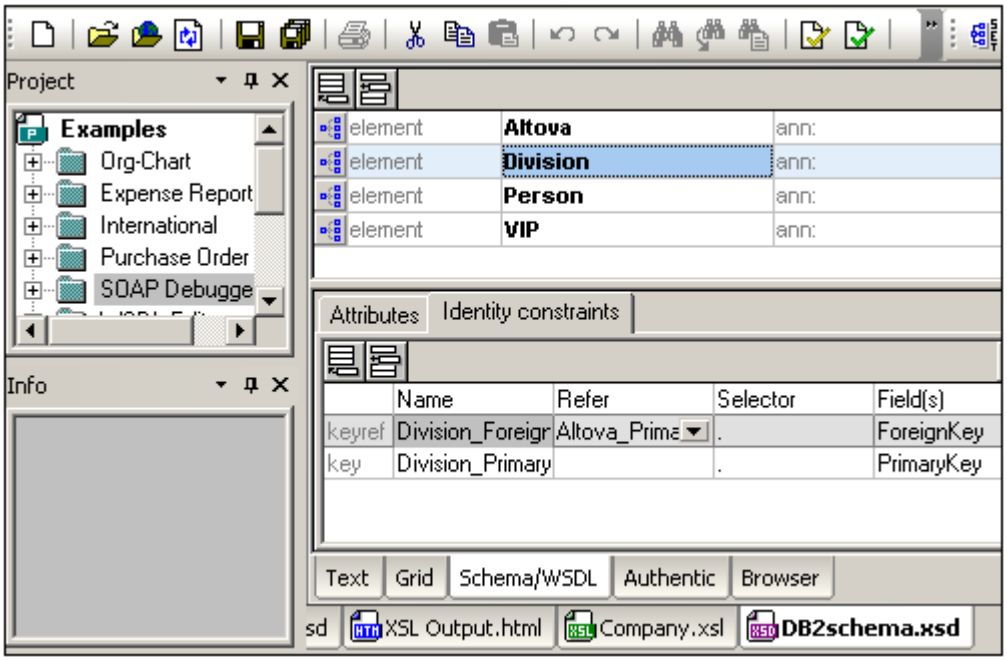
Alle Einstellungen können unter **Projekt | Projekteigenschaften...** definiert werden. Im nächsten Abschnitt werden Sie über das Menü "Projekt" ein Projekt erstellen.

Des Weiteren stehen die folgenden Projektfunktionen zur Verfügung:

- XML-Dateien können unter Versionskontrolle gestellt werden (**Projekt | Versionskontrolle | Zu Versionskontrolle hinzufügen...**). Weitere Informationen erhalten Sie im Abschnitt Versionskontrolle der Online-Hilfe.
- Persönliche, netzinterne und Webordner können in Projekte eingefügt werden, welches die Batchvalidierung ermöglicht.

Erstellen eines Projekts

Im Hauptfenster sind nun viele verschiedene Dateien offen.

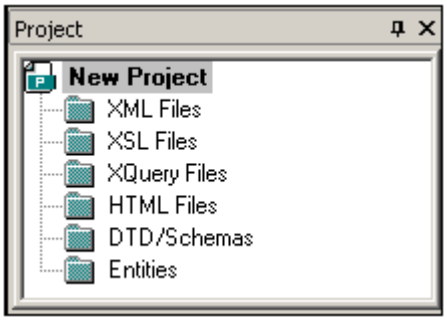


Sie können diese Dateien in einem Tutorial-Projekt gliedern. Zuerst müssen Sie ein neues Projekt erstellen und anschließend die Tutorial-Dateien zu den jeweiligen Unterordnern des Projekts hinzufügen.

Erstellen eines einfachen Projekts

Um ein einfaches Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Projekt | Neu**. Im Projektfenster wird ein Projektordner mit dem Namen `New Project` erstellt. Das neue Projekt enthält leere Ordner für typische XML-Dateikategorien in einem Projekt (*siehe Abbildung unten*).



2. Klicken Sie auf das Register `CompanyLast.xml` um diese Datei im Hauptfenster aktiv zu setzen.
3. Wählen Sie die Menüoption **Projekt | Aktive und verwandte Dateien hinzufügen**. Es werden zwei Dateien zum Projekt hinzugefügt: `CompanyLast.xml` und `AddressLast.xsd`. Beachten Sie, dass durch Verarbeitungsanweisungen referenzierte Dateien (wie z.B. XSLT-Dateien) nicht als verwandte Dateien gelten.
4. Wählen Sie die Menüoption **Projekt | Projekt speichern** und speichern Sie das Projekt unter dem Namen `Tutorial`.

Hinzufügen von Dateien zu einem Projekt
Sie können auch andere Dateien zum Projekt hinzufügen. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie auf eine beliebige offene XML-Datei (mit der Erweiterung `.xml`) mit Ausnahme von `CompanyLast.xml`, um diese XML-Datei zur aktiven zu machen. (Wenn keine andere XML-Datei geöffnet ist, öffnen Sie eine oder erstellen Sie eine neue XML-Datei).
2. Wählen Sie die Menüoption **Projekt | Aktive Datei hinzufügen.** Die XML-Datei wird anhand ihres Dateityps (.xml) in den Ordner XML Files hinzugefügt.
3. Fügen Sie nun auf dieselbe Art eine HTML-Datei und eine XSD-Datei (z.B. `Company.html` und `DB2schema.xsd`) zum Projekt hinzu. Diese Dateien werden zum Ordner HTML Files bzw. DTD/Schemas hinzugefügt.
4. Speichern Sie das Projekt, entweder durch Auswahl der Menüoption **Projekt | Speichern** oder durch Auswahl einer beliebigen Datei oder eines Ordners im Projektfenster und Klicken auf das Symbol "Speichern" in der Symbolleiste (oder **Datei | Speichern**).

Hinweis: Alternativ dazu können Sie mit der rechten Maustaste auf einen Projektordner klicken und den Befehl "Aktive Datei hinzufügen" auswählen, um die aktive Datei zu diesem bestimmten Ordner hinzuzufügen.

Andere nützliche Befehle
Hier sind einige weitere gebräuchliche Projektbefehle aufgelistet:

- Um einen neuen Ordner zu einem Projekt hinzuzufügen, wählen Sie die Menüoption **Projekt | Projektordner in Projekt einfügen** und fügen Sie den Namen des Projektordners ein.
- Um einen Ordner aus einem Projekt zu löschen, rechtsklicken Sie auf den Ordner und wählen Sie im Kontextmenü den Befehl **Löschen**.
  Um eine Datei aus einem Projekt zu löschen, wählen Sie die Datei aus und drücken Sie die **Entf**-Taste.

2.1.9 Fertig!

Herzlichen Glückwunsch!

Wir hoffen, dass Ihnen diese Anleitung geholfen hat, um die Grundzüge von XMLSpy 2005 kennen zu lernen. Weitere Informationen finden Sie in der kontextsensitiven Online-Hilfe. Oder drucken Sie die PDF-Version des Tutorials aus, die sich im Anwendungsordner von XMLSpy 2005 befindet (tutorial.pdf).

2.2 Tutorial Authentic-Ansicht

In der Authentic-Ansicht können Sie eine bereits vorhandene XML-Datei, die mit einem StyleVision Power Stylesheet verknüpft ist, öffnen. Anschließend können Sie die Datei mit Hilfe der verschiedenen Funktionen der Authentic-Ansicht ändern. Dieses Tutorial gliedert sich in drei grundlegende Teile:

- Öffnen einer bestehenden XML-Datei in der Authentic-Ansicht
- Bearbeiten von Daten (Hinzufügen von neuen Dokumentenkomponenten sowie Inhalt); dieser Abschnitt nimmt den Großteil dieses Tutorials ein.
- Drucken des Dokuments

Denken Sie daran, dass dieses Tutorial als Einführung gedacht ist und daher nicht auf komplizierte Aspekte eingeht. Zusätzliches Referenzmaterial und nähere Informationen zu einzelnen Funktionen finden Sie in den Abschnitten zur Oberfläche der Authentic-Ansicht.

Voraussetzungen für das Tutorial
Sie finden alle für das Tutorial benötigten **Dateien** im Ordner `Examples` des Altova-Applikationsordners. Es sind die Dateien:

- `NanonullOrg.xml` (das XML-Dokument, das Sie öffnen)
- `NanonullOrg.sps` (das StyleVision Power Stylesheet, mit dem das XML-Dokument verknüpft ist)
- `NanonullOrg.xsd` (das XML-Schema, auf dem das XML-Dokument und das StyleVision Power Stylesheet basieren und mit dem diese verknüpft sind)
- `nanonull.gif` und `Altova_right_300.gif` (zwei Bilddateien, die in dem Tutorial verwendet werden)

Hinweis: An einigen Stellen des Tutorials werden Sie gebeten, sich den XML-Text des XML-Dokuments anzusehen (also nicht die Authentic-Ansicht des Dokuments). Wenn die von Ihnen verwendete Altova Produktversion keine Textansicht enthält (dies ist bei der kostenlosen Authentic 2005 Desktop Edition und der Authentic 2005 Browser Edition der Fall), verwenden Sie einen einfachen **Texteditor**, wie Wordpad oder Notepad, um den Text des XML-Dokuments anzuzeigen.

Vorsicht: Verwenden Sie für das Tutorial am besten eine Kopie der Datei `NanonullOrg.xml`, damit Sie das Original im gegebenen Fall jederzeit wieder aufrufen können.

2.2.1 Öffnen eines XML-Dokuments in der Authentic-Ansicht

Die Datei `NanonullOrg.xml` befindet sich im Installationsordner der Applikation im Ordner `Examples`. Normalerweise lautet der Pfad:
`c:\Program\Files\Altova\XMLSpy 2005\Examples\NanonullOrg.xml`

Die Datei `NanonullOrg.xml` lässt sich auf zwei Arten öffnen:

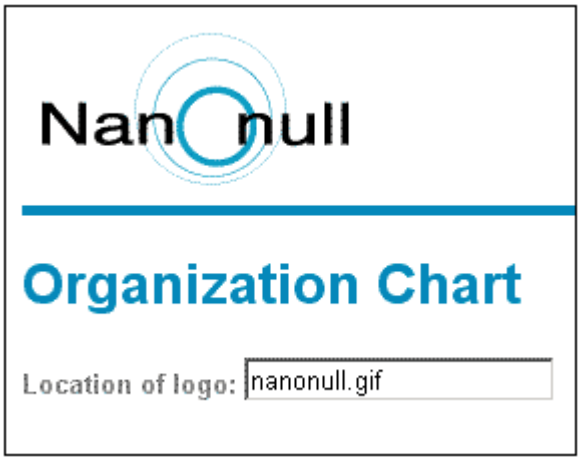
- Klicken Sie in Ihrem Altova-Produkt auf **Datei | öffnen**, suchen Sie anschließend die Datei `NanonullOrg.xml` im Dialogfeld, das nun angezeigt wird und klicken Sie auf "Öffnen".
- Suchen Sie die Datei im Windows Explorer, rechtsklicken Sie darauf und wählen Sie Ihr Altova-Produkt aus.

Die Datei `NanonullOrg.xml` wird direkt in der Authentic-Ansicht geöffnet, weil:

1. der Datei bereits ein StyleVision Power Stylesheet zugewiesen wurde.
2. im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Ansicht" die Option, XML-Dateien in der Authentic-Ansicht zu öffnen, wenn eine SPS-Datei zugewiesen

 © 2005 Altova GmbH

wurde, aktiviert ist. (Andernfalls würde die Datei in der Textansicht geöffnet.)



Beachten Sie: Wie ein XML-Dokument in der Authentic-Ansicht angezeigt wird, wird im StyleVision Power Stylesheet definiert. Ohne ein StyleVision Power Stylesheet kann das Dokument nicht in der Authentic-Ansicht dargestellt werden.

Hinweis: Alternativ dazu könnten Sie eine XML-Vorlage in der Authentic-Ansicht öffnen, indem Sie ein StyleVision Power Stylesheet auswählen. Gehen Sie dazu folgendermaßen vor:

1. Wählen Sie **Datei | Neu** und wählen Sie im Dialogfeld "Neues Dokument anlegen" XML als den Dateityp der neu zu erstellenden Datei aus.
2. Klicken Sie auf **STYLEVISION Stylesheet auswählen...** und suchen Sie das gewünschte StyleVision Power Stylesheet.

Wenn dem StyleVision Power Stylesheet eine XML-Vorlagendatei zugewiesen wurde, werden die Daten in der XML-Vorlagendatei als Ausgangsdaten der in der Authentic-Ansicht erstellten Vorlage verwendet.

2.2.2 Eingeben von Daten in der Authentic-Ansicht

Die Eingabe von Daten in der Authentic-Ansicht wird vom damit verknüpften StyleVision Power Stylesheet gesteuert.

Im **einfachsten Szenario** geben Sie Inhalt als frei fließenden Text oder in Dateneingabefelder ein, oder Sie wählen die Einträge aus einer Liste von Benutzeroptionen aus. Kurz und gut Sie konzentrieren sich auf die Eingabe von Inhalt; benutzerseitige Änderungen an der Struktur des Dokuments (Hinzufügen von Elementen, Tabellen usw.) und die Formatierung des Dokuments werden auf ein Minimum beschränkt. Dies stellt sicher, dass das Dokument gültig ist und die Daten richtig sind. Sie können sich somit auf den Inhalt konzentrieren.

In den meisten Fällen haben Sie die Möglichkeit, **einige Elemente hinzuzufügen**. Entweder dies geschieht implizit, wenn Sie z.B. die Eingabetaste drücken, um ein neues Absatzelement hinzuzufügen oder auf ein Symbol klicken, um den Text fett zu formatieren. Oder es geschieht explizit, wenn Sie z.B. ein Element über die Element-Eingabehilfe hinzufügen. Im zweiten Szenario müssen Sie wissen, wie das Dokument strukturiert ist.

Das andere Extrem ist, wenn Sie als Benutzer vollkommene Freiheit beim Strukturieren des Dokuments haben. Sie benötigen dazu allerdings eine gute Kenntnis des Schemas, auf dem das Dokument basiert.

In der Authentic-Ansicht können Sie die folgenden Arten von Daten und Datenstrukturen eingeben oder bearbeiten:

- Elementinhalt (kann als Text oder über ein Dateneingabe-Steuerelement eingegeben werden)
- Attributwerte (können als Text oder über ein Dateneingabe-Steuerelement oder als Wert in der Attribut-Eingabehilfe eingegeben werden)
- Entities (können über die Entities-Eingabehilfe eingefügt werden)
- Elemente (können hinzugefügt, in andere Elemente geändert und gelöscht werden)
- XML-Tabellen (können eingefügt werden, und die Struktur, Formatierung und der Inhalt können definiert werden)

In diesem Tutorial lernen Sie, wie man in der Authentic-Ansicht Elemente bearbeitet und Inhalt eingibt. XML-Tabellen werden im Abschnitt Tabellen in der Authentic-Ansicht näher behandelt.

2.2.3 Hinzufügen von Dokumentinhalt und Elementen

Hinzufügen von Textinhalt
Sie können Elementinhalt und Attributwerte direkt als Text eingeben. Setzen Sie dazu den Cursor an die Stelle, an der Sie den Text eingeben möchten und tippen Sie ihn ein. Sie können Inhalt auch durch Markieren von Text und Eingabe des Ersetzungstexts eingeben bzw. den markierten Text löschen.

Um den Namen der Firma von Nanonull, Inc in Nanonull USA, Inc zu ändern, platzieren Sie den Cursor hinter Nanonull und geben Sie den einzufügenden Text ein.

Wenn Text editierbar ist, können Sie den Cursor hineinsetzen und den Text markieren, andernfalls ist das nicht der Fall. Versuchen Sie, einen der Feldnamen "Street", "City", "State/Zip" (im Adressblock) zu ändern. Sie können den Cursor nicht in diesen Text platzieren, da dies kein XML-Inhalt ist, sondern aus dem StyleVision Power Stylesheet stammt.

Hinweis: Im Markup-Ausblendemodus können leere Elemente leicht übersehen werden. Um sicherzustellen, dass Sie keine leeren Elemente übersehen, wechseln Sie in den Modus "Große Markup-Symbole einblenden" oder "Kleine Markup-Symbole einblenden".

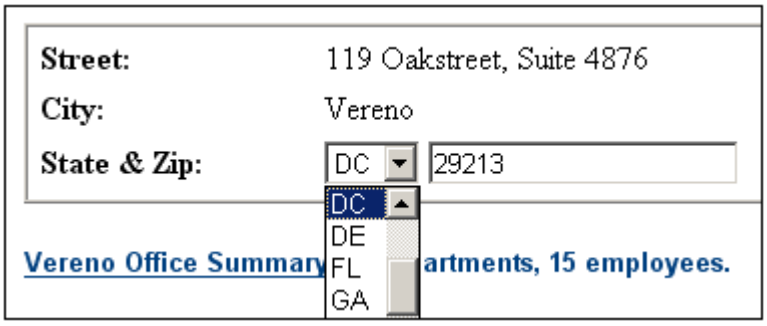
Hinzufügen von Inhalt über ein Dateneingabe-Steuerelement
Beim Editieren von Inhalt wird dieser direkt durch Eingabe des Texts als Elementinhalt eingegeben. Es gibt aber noch eine andere Methode, um **Elementinhalt** (oder Attributwerte) in der Authentic-Ansicht einzugeben: mit Hilfe von Dateneingabe-Steuerelemente.

Nachfolgend sehen Sie eine Liste von Dateneingabe-Steuerelementen in der Authentic-Ansicht sowie eine Erklärung dazu, wie Daten bei jedem dieser Elemente in die XML-Datei eingegeben werden.

© 2005 Altova GmbH

Dateneingabe-Steuerelement	Daten in der XML-Datei
Eingabefeld (Textfeld)	Text wird vom Benutzer eingegeben
Eingabefeld mit mehreren Zeilen	Text wird vom Benutzer eingegeben
Dropdown-Liste	Auswahl des Benutzers wird auf einen Wert gemappt
Kontrollkästchen	Auswahl des Benutzers wird auf einen Wert gemappt
Optionsfeld	Auswahl des Benutzers wird auf einen Wert gemappt
Schaltfläche	Auswahl des Benutzers wird auf einen Wert gemappt

In der statischen Tabelle mit den Adressfeldern (siehe unten) stehen zwei Dateneingabe-Steuerelemente zur Verfügung: Ein Eingabefeld für das Feld "Zip" (PLZ) und eine Dropdown-Liste für das Feld "State" (Bundesstaat). Die Werte, die Sie in die Textfelder eingeben werden direkt als XML-Inhalt der entsprechenden Elemente eingegeben. Bei anderen Dateneingabe-Steuerelementen wird Ihre Auswahl auf einen Wert gemappt.



Hier sehen Sie den XML-Text für die oben gezeigte Authentic-Ansicht:

```
<Address>
  <ipo:street>119 Oakstreet, Suite 4876</ipo:street>
  <ipo:city>Vereno</ipo:city>
  <ipo:state>DC</ipo:state>
  <ipo:zip>29213</ipo:zip>
</Address>
```

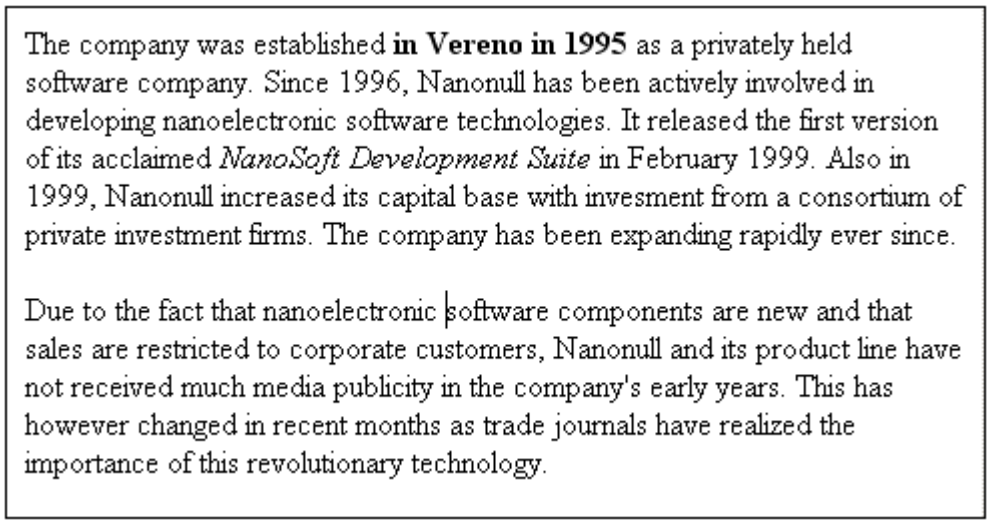
Beachten Sie, dass die Dropdown-Listen-Auswahl "DC" auf einen Wert "DC" gemappt ist. Der Wert im Feld "zip" wird direkt als Inhalt des Elements `ipo:zip` eingegeben.

Hinzufügen von Elementen = Einfügen oder Anhängen
Sie können ein Element durch **Einfügen vor** oder **Anhängen nach** dem **aktuellen Element** (dem Element, in dem sich der Cursor befindet) hinzufügen.

Fügen Sie einen weiteren Absatz zur Firmenbeschreibung hinzu. Dazu müssen Sie ein neues Absatzelement (in diesem Fall `para`) anhängen und Inhalt für das Element eingeben. Gehen Sie dabei vor wie folgt:

1. Setzen Sie den Cursor in den letzten Absatz des Beschreibungstexts.

2. Klicken Sie auf das Symbol (vor Element einfügen), das sich in der Element-Eingabehilfe links vom Element `para` befindet. Daraufhin wird ein `para`-Element **vor** dem aktuellen `para` eingefügt, was wir in diesem Fall nicht wollen.
3. Klicken Sie auf das Symbol (Element anhängen), das sich in der Element-Eingabehilfe links vom Element `para` befindet. Daraufhin wird - wie in diesem Fall gewünscht - ein `para`-Element angehängt.
4. Geben Sie den Inhalt von `para` an der blinkenden Cursorposition ein.



Alternativ dazu können Sie zum Anhängen eines `para`-Elements (oder eines beliebigen Absatztyps) an der Stelle, an der Sie das neue Element einfügen möchten, die **Eingabetaste** drücken.

Hinweis: Mit der Eingabetaste können Sie Listenelemente auch in nummerierten Listen und Aufzählungen einfügen und anhängen.

Hinzufügen von Elementen im Dokument
Beachten Sie bitte, dass Sie **nur Elemente derselben oder einer übergeordneten Ebene** vor oder nach dem aktuellen Element einfügen oder anhängen können.

Elemente derselben Ebene sind Geschwisterelemente. Geschwisterelemente eines Absatzelements wären andere Absatzelemente, es könnte sich aber auch um Listen, eine Tabelle, eine Grafik usw. handeln. Geschwisterelemente können vor oder nach dem Element vorkommen.

Elemente übergeordneter Ebenen sind Vorfahren-Elemente und Geschwisterelemente von Vorfahren. Bei Vorfahren eines Absatzelements könnte es sich um einen Abschnitt, ein Kapitel, einen Artikel usw. handeln. Ein Absatz in einem gültigen XML-Dokument hätte bereits Vorfahren. Wenn Sie daher in der Authentic-Ansicht eine Element einer übergeordneten Ebene erstellen, wird dieses als Geschwisterelement des jeweiligen Vorfahrenelements erstellt. Wenn z.B. ein Abschnittselement an einen Absatz angehängt wird, wird es als Geschwisterelement des Abschnitts erstellt - und zwar als der letzte Abschnitt auf dieser Ebene.

An jeder Stelle im Dokument werden die Elemente, die sie einfügen oder anhängen können, mit den dazugehörigen Symbolen links davon in der Element-Eingabehilfe angezeigt.

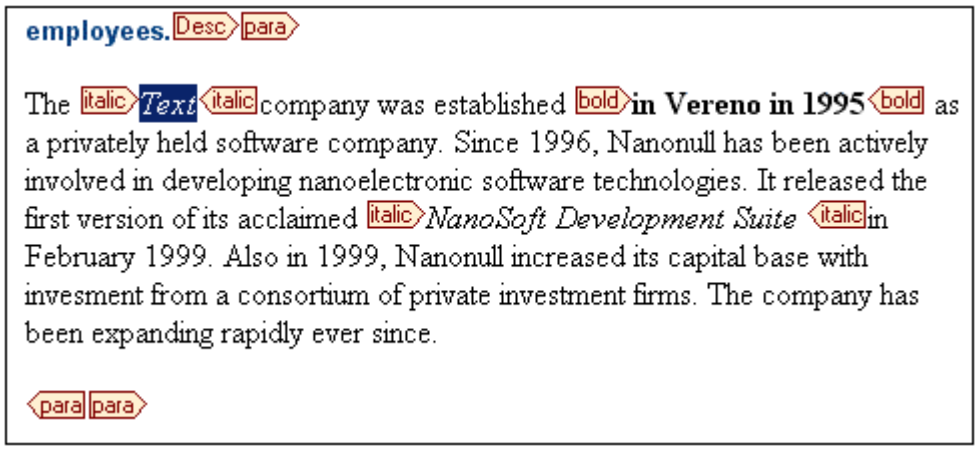
 © 2005 Altova GmbH

2.2.4 Einfügen eines Elements

Sie können ein Element als **Child-Element** des **aktuellen Elements** (also des Elements, in dem sich der Cursor befindet) **einfügen.**

In `NanonullOrg.xml` kann das Element `para` die Elemente `italic` und `bold` enthalten. Experimentieren Sie nun mit dem Einfügen von Elementen:

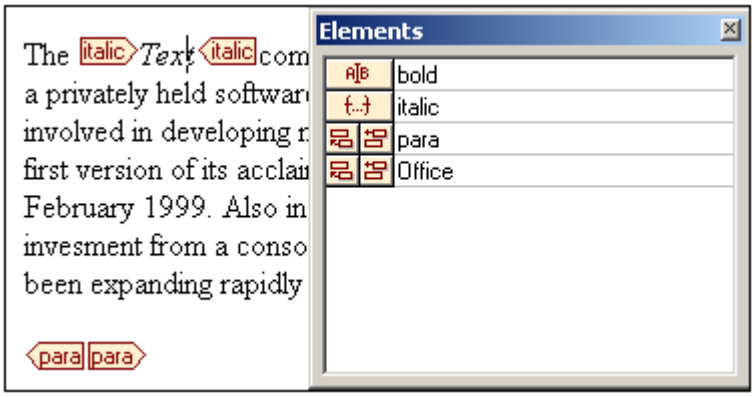
1. Setzen Sie den Cursor an eine Stelle im Element `para`, an der Sie das Child-Element einfügen möchten. (Überprüfen Sie die Statusleiste am linken unteren Rand, um zu sehen, wo Sie sich im Dokument befinden). Beachten Sie, dass die Elemente `italic` und `bold` in der Eingabehilfe mit einem Symbol (Element einfügen) neben dem Element aufgelistet sind.
2. Klicken Sie auf das Symbol neben dem Element `italic`. Daraufhin wird das Element `italic` mit einem Platzhalter "Text", der markiert erscheint, eingefügt. Wechseln Sie in den Modus "Große Markup-Symbole einblenden", um die Element-Tags zu sehen.



3. Geben Sie den Inhalt des Elements ein.

Wenn sich der Cursor innerhalb eines Elements befindet, werden alle Elemente, die als Child-Elemente eingefügt werden können, in der Element-Eingabehilfe mit einem "Element einfügen"-Symbol daneben angezeigt.

Setzen Sie den Cursor nun in das Element `italic` und sehen Sie sich die Element-Eingabehilfe an.



Die Element-Eingabehilfe zeigt an, dass das Element `bold` als Child-Element von `italic` eingefügt werden kann. Klicken Sie auf das "Element einfügen"-Symbol für das Element `bold`, um zu sehen was passiert.

2.2.5 Anwenden oder Löschen von Elementen

Anwenden (Ersetzen) eines Elements

Mit Hilfe des Symbols "Element anwenden" können Sie das ausgewählte Element durch ein anderes an dieser Stelle zulässiges Element ersetzen.

Um das Element `bold` auf das soeben erstellte Element `italic` anzuwenden, gehen Sie folgendermaßen vor:

1. Wechseln Sie in den Modus "Große Markup-Symbole einblenden". In diesem Modus ist es einfacher, ein ganzes Element auszuwählen.
2. Wählen Sie das Element `italic`, indem Sie entweder auf seinen Start- oder seinen End-Tag klicken. Das Element `bold` wird in der Eingabehilfe mit dem Symbol (Element anwenden) daneben angezeigt.
3. Klicken Sie auf das Symbol. Das Element `italic` ändert sich im Hauptfenster in `bold`.

Wenn das gesamte Element ausgewählt ist, werden Elemente, die auf das ausgewählte Element angewendet werden können, in der Element-Eingabehilfe mit dem Symbol (Element anwenden) daneben angezeigt.

Hinweis: Es ist wichtig, dass Sie das ganze Element und nicht nur seinen Inhalt auswählen. Wenn nur der Inhalt ausgewählt wird, wird der "Element löschen"-Kontext des markierten Elements aktiviert.

Löschen eines Elements

Wenn Sie ein Element löschen, entfernen Sie seinen Markup-Code, ohne den Inhalt zu ändern.

Um das Element `italic` zu löschen, gehen Sie vor wie folgt:

1. Wechseln Sie in den Modus "Große Markup-Symbole einblenden."
2. Setzen Sie den Cursor als Einfügepunkt in das Element `italic`. Das Element `italic` wird in der Element-Eingabehilfe mit dem Symbol daneben angezeigt.
3. Klicken Sie auf das Symbol . Das Element `italic` wird im Hauptfenster entfernt und die Kursivformatierung wird entfernt.

Sie können auch ein Textfragment im Element `italic` auswählen (anstatt den Cursor als Einfügepunkt zu setzen) und das Element `italic` löschen. Der einzige Unterschied ist, dass das Element `italic` in der Element-Eingabehilfe mit einer zweiten Variante des "Element löschen"-Symbols daneben angezeigt wird: . Dieses Symbol hat dieselbe Funktion wie das Symbol .

Hinweis: Der Cursor muss unbedingt in den Element-Inhalt platziert werden. Es darf nicht das ganze Element ausgewählt sein, da sonst der "Element anwenden"-Kontext aktiviert wird, der zum Ersetzen des ausgewählten Elements durch ein anderes dient.

2.2.6 Eingabe von Attributwerten

Ein Attribut ist eine Eigenschaft eines Elements. Ein Element kann beliebig viele Attribute haben. Attribute haben Werte. Manchmal müssen XML-Daten als Attributwert eingegeben werden. In der Authentic-Ansicht werden Attributwerte auf zwei Arten eingegeben:

© 2005 Altova GmbH

- Als Inhalt im Hauptfenster, wenn das Attribut so definiert wurde, dass es die Eingabe des Wert auf diese Art zulässt.
- In der Attribut-Eingabehilfe

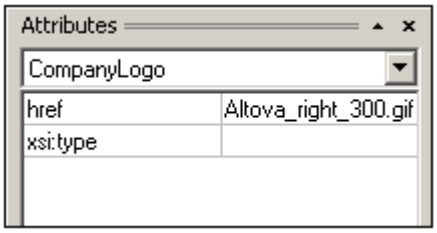
Attributwerte im Hauptfenster
Attributwerte können als Fließtext oder als Text in einem Eingabefeld eingegeben werden oder als vom Benutzer aus einer Auswahl gewählter Eintrag, der auf einen XML-Wert gemappt wird. Attributwerte werden auf dieselbe Art eingegeben, wie Element-Inhalt: siehe Hinzufügen von Dokumentinhalt und Elementen.

In solchen Fällen erfolgt die Unterscheidung zwischen Element-Inhalt und Attributwerten durch das StyleVision Power Stylesheet und die Daten werden entsprechend behandelt.

Attributwerte in der Attribut-Eingabehilfe
Wenn Sie einen Attributwert eingeben oder ändern wollen, können Sie dies in der Attribut-Eingabehilfe tun.

Der Pfad des in `NanonullOrg.xml` verwendeten Logos wird als der Wert des `href` Attributs des Elements `CompanyLogo` gespeichert. Um ein anderes Logo zu verwenden, gehen Sie vor wie folgt:

1. Wählen Sie das Element `CompanyLogo`, indem Sie entweder das Element auswählen oder auf das Nanonull-Logo klicken. Die Attribute des Elements `CompanyLogo` werden in der Attribut-Eingabehilfe angezeigt.
2. Ändern Sie in der Attribut-Eingabehilfe den Wert des Attributs `href` von `nanonull.gif` in `Altova_right_300.gif` (ein Bild im Ordner `Examples`).



Daraufhin wird das Nanonull-Logo durch das Altova-Logo ersetzt.

Hinweis: Wenn der Wert eines Attributs eingegeben muss, erstellt der Autor des StyleVision Power Stylesheet normalerweise einen Mechanismus für die Eingabe dieser Daten in der Authentic-Ansicht.

2.2.7 Hinzufügen von Entities

Bei einer Entity in der Authentic-Ansicht handelt es sich normalerweise (nicht notwendigerweise) um XML-Daten wie z.B. ein einzelnes Zeichen, einen Textstring oder auch ein Fragment eines XML-Dokuments. Auch eine Binärdatei wie z.B. eine Bilddatei kann eine Entity sein.

Alle verfügbaren Entities werden in der Entities-Eingabehilfe angezeigt. Um eine Entity einzufügen, doppelklicken Sie darauf.

Ändern Sie in `NanonullOrg.xml` den Jobtitel von Joe Martin (in Marketing) in Marketing Manager Europe & Asia.

Das kaufmännische Und (&) hat in XML eine spezielle Bedeutung (wie auch das Apostroph, das "Größer als"- und das "Kleiner als"-Zeichen und das doppelte Anführungszeichen). Um diese Zeichen einzufügen, werden Entities verwendet, damit sie nicht mit Zeichen, die eine XML-Bedeutung haben, verwechselt werden. In `NanonullOrg.xml` wurden für diese Zeichen Entities deklariert und in der Entities-Eingabehilfe angezeigt.

So fügen Sie in den Titel "Marketing Manager Europe & Asia" eine Entity für ein kaufmännisches Und ein:

1. Platzieren Sie den Cursor dorthin, wo das kaufmännische Und eingefügt werden soll.
2. Doppelklicken Sie auf die Entity, die als "amp" aufgelistet ist.

Daraufhin wird ein kaufmännisches Und (Englisch: Ampersand) eingefügt.

Hinweis: Die Entities-Eingabehilfe ist nicht kontextsensitiv. Es werden immer alle verfügbaren Entities angezeigt, egal wo der Cursor sich gerade befindet. Dies bedeutet nicht, dass eine Entity überall im Dokument eingefügt werden kann. Wenn Sie sich nicht sicher sind, validieren Sie das Dokument nach dem Einfügen der Entity mit: **XML | Validieren** (**F8**).

Siehe auch: Attribut-Eingabehilfe unter Eingabehilfen der Authentic-Ansicht.

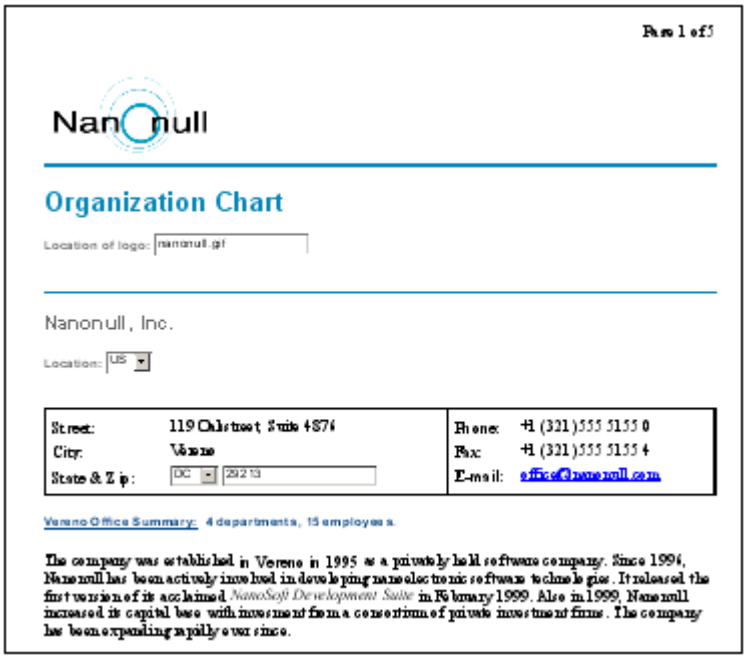
2.2.8 Drucken des Dokuments

Bei Ausdruck eines XML-Dokuments von der Authentic-Ansicht aus bleibt die in der Authentic-Ansicht sichtbare Formatierung erhalten.

So drucken Sie die Datei `NanonullOrg.xml`:

1. Wechseln Sie in den Modus "Markup-Symbole ausblenden", wenn Sie gerade einen anderen Modus verwenden, da sonst auch die Markup-Symbole gedruckt werden.
2. Wählen Sie **Datei | Druckansicht**, um eine Vorschau aller Seiten anzuzeigen. In der Abbildung unten sehen Sie einen Teil einer um 50 % verkleinerten Seite in der Druckansicht.

© 2005 Altova GmbH



Beachten Sie, dass die Formatierung auf der Seite dieselbe ist, wie in der Authentic-Ansicht.

3. Um die Datei zu drucken, wählen Sie **Datei | Drucken**.

Beachten Sie, dass Sie auch eine Version des Dokuments drucken können, in der (kleine) Markup-Symbole angezeigt werden. Wechseln Sie dazu in den Modus "Kleine Markup-Symbole einblenden" oder "Große Markup-Symbole einblenden" und wählen Sie den Befehl "Drucken". In beiden Modi werden im Ausdruck kleine Markup-Symbole angezeigt.

2.3 WSDL Tutorial

Ziel dieses Tutorials ist es:

- ein WSDL-Dokument in der graphischen WSDL Design-Ansicht (Schema/WSDL-Ansicht) von Grund auf neu zu erstellen
- eine Verbindung zu einem Webservice herzustellen, die WSDL-Datei lokal zu speichern und einen SOAP Request an den Webservice zu senden

Sie können das alles in der graphischen WSDL Design-Ansicht tun und benötigen die Textansicht hierfür nicht. Sie können die WSDL-Komponenten direkt mit Hilfe von Drag & Drop verschieben und Eigenschaftswerte in den Eingabehilfen der WSDL Design-Ansicht eingeben.

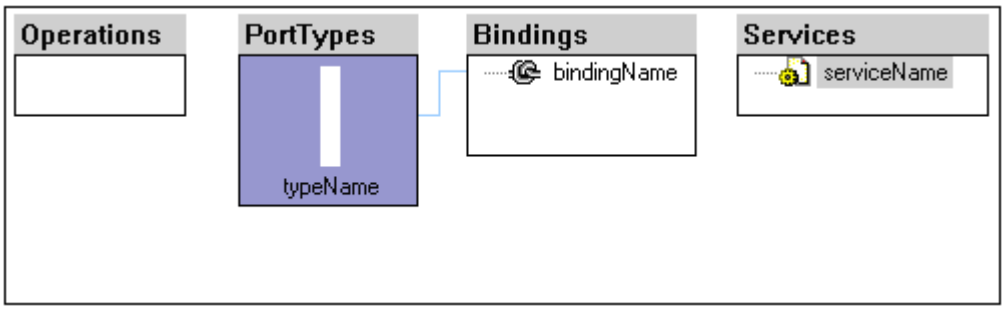
Gehen Sie dazu vor wie folgt:

- Erstellen Sie mit Hilfe des Menübefehls **Datei | Neu** ein rudimentäres WSDL-Dokument
- Weisen Sie dem Service Ports zu
- Weisen Sie einem Binding eine Verbindungsadresse zu
- Fügen Sie Input und Output Messages sowie die dazugehörigen Parameter hinzu
- Fügen Sie Messages zu Operations hinzu
- Definieren Sie den Typ des Binding
- Validieren Sie die WSDL-Datei, um ihren Status zu überprüfen und ergänzen Sie die WSDL-Datei anhand der eventuell angezeigten Eingabeaufforderungen

2.3.1 Erstellen eines Service und Port

Erstellen eines Service und eines damit verknüpften Port:

1. Wählen Sie **Datei | Neu** und anschließend den Eintrag **WSDL Web Service Description Language** und bestätigen Sie mit OK.

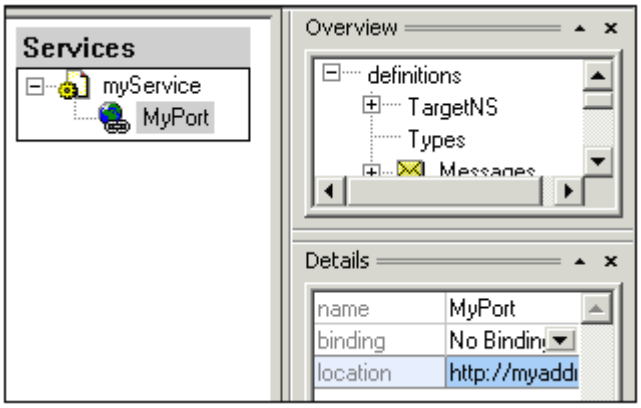


   Daraufhin wird das Grundgerüst einer WSDL-Datei erstellt, das automatisch in der graphischen WSDL-Ansicht geöffnet wird.
2. Klicken Sie im Fenster "Services" auf den Eintrag **serviceName** und ändern Sie ihn in einen Service-Namen Ihrer Wahl, z.B. MyService (oder klicken Sie auf den Eintrag, drücken Sie F2 und geben Sie den neuen Namen ein).
3. Rechtsklicken Sie auf **MyService** und wählen Sie im Popup-Menü den Eintrag **Append Port**.
   Daraufhin wird zum Fenster "Services" ein Eintrag "NewPort" hinzugefügt.
4. Klicken Sie darauf und benennen Sie ihn um, z.B. in MyPort.

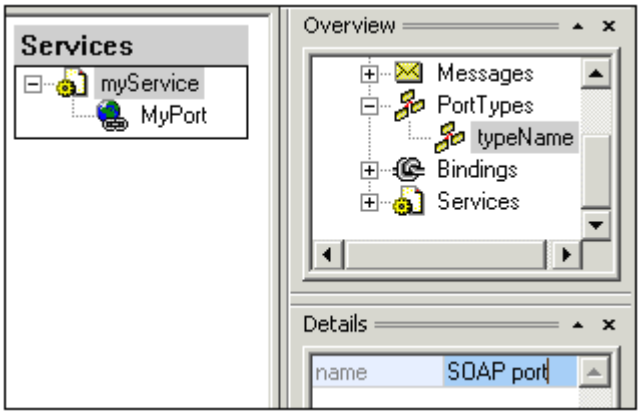
5. Doppelklicken Sie, während der Eintrag **MyPort** aktiv ist, in der Details-Eingabehilfe auf

 © 2005 Altova GmbH

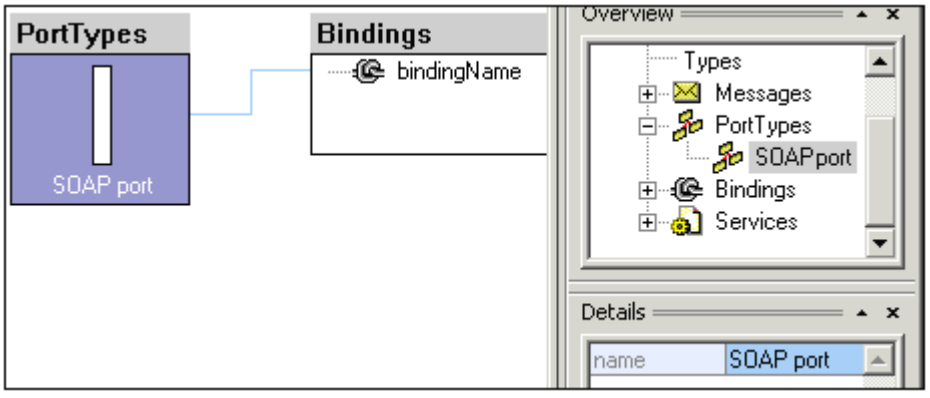
das Feld **Location** und geben Sie die **Verbindungsadresse** für diesen Service ein, z.B. **http://myaddress.com**.



6. Klicken Sie im Fenster **Overview** unter PortTypes auf den Eintrag **typeName** und benennen Sie ihn im Feld **Name** der **Details-**Eingabehilfe in SOAPport um.

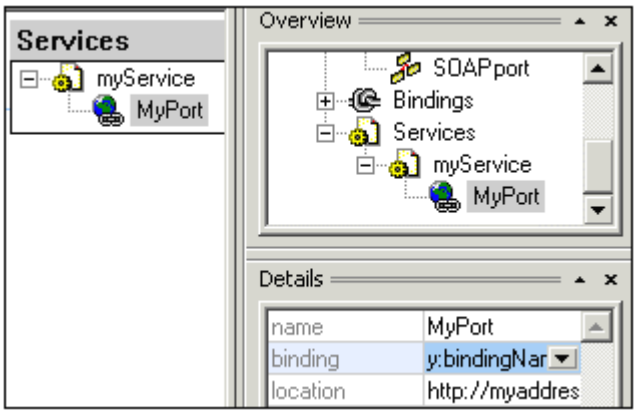


Dadurch wird der Name des PortType in allen damit in Zusammenhang stehenden PortType-Fenstern geändert.

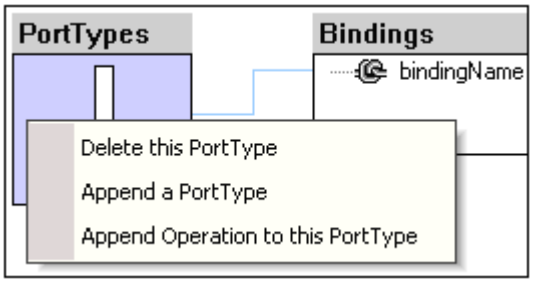


Verknüpfen einer Verbindungsadresse mit einem Binding (Definieren einer Operation für einen PortType):

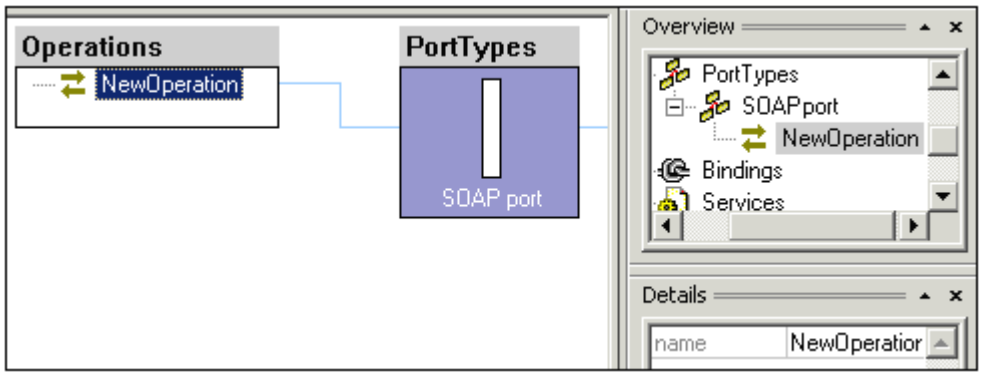
1. Klicken Sie, während der Eintrag **MyPort** aktiv ist, in der Details-Eingabehilfe auf die **Binding-**Dropdown-Liste und wählen Sie den zur Verfügung stehenden Eintrag (**y:bindingName**).



2. Rechtsklicken Sie auf das Fenster **PortTypes** und wählen Sie den Befehl **Append Operation to this PortType**.



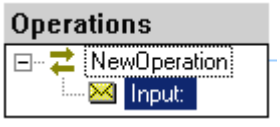
Daraufhin wird im Fenster "Operations" eine neue Operation erstellt, die automatisch zum aktuellen Binding im Fenster "Bindings" hinzugefügt wird.



2.3.2 Hinzufügen von Input- und Output-Messages und Parametern

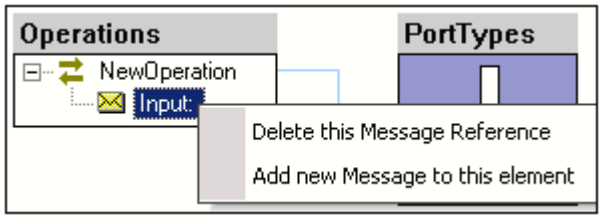
Hinzufügen von Input- und Output-Messages zu einer Operation:

1. Rechtsklicken Sie im Fenster **Operations** auf den Eintrag **NewOperation** und wählen Sie den Befehl **Add Input element**.

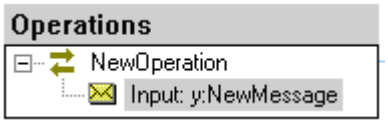


2. Rechtsklicken Sie auf den Eintrag **Input:** und wählen Sie den Befehl **Add new Message to this element**.

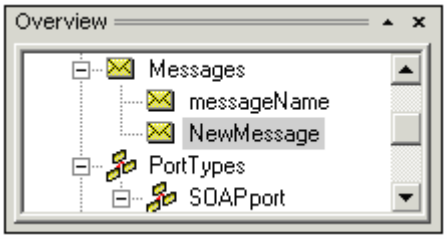
© 2005 Altova GmbH



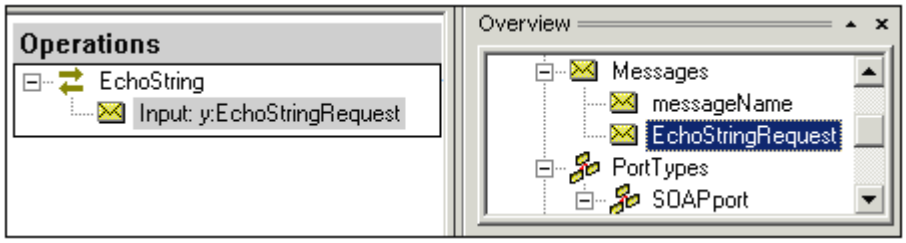
Daraufhin wird automatisch "y:NewMessage" zu diesem Input-Eintrag hinzugefügt.



3. Klicken Sie auf den Eintrag **NewOperation** und benennen Sie ihn z.B. in EchoString um.
4. Klicken Sie in der Eingabehilfe **Overview** auf **NewMessage** und benennen Sie den Eintrag z.B. in EchoStringRequest um.

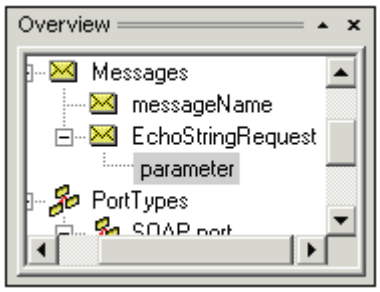


Dadurch wird der Eintrag im Fenster "Operations" automatisch umbenannt.



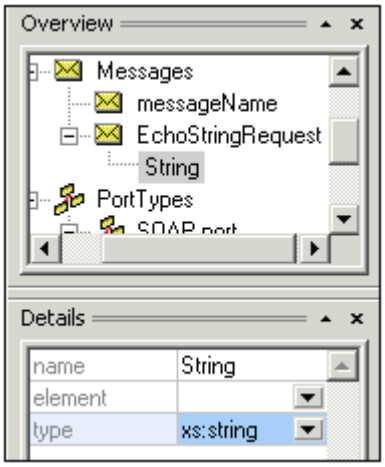
Hinzufügen von Parametern zu Messages:

1. Rechtsklicken Sie in der Eingabehilfe **Overview** auf **EchoStringRequest** und wählen Sie den Befehl **Add Message Part (Parameter)**.
   Es wird dadurch ein Parameter zum EchoStringRequest hinzugefügt.



2. Klicken Sie im Fenster "Overview" auf den neu hinzugefügten Message Part und benennen Sie ihn z.B. in String um.
3. Gehen Sie in der **Details-**Eingabehilfe zur Dropdown-Liste **type** und geben Sie

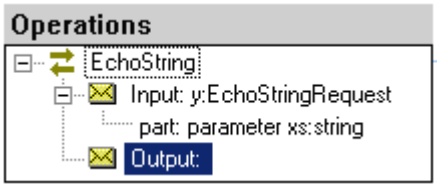
xs:string ein oder wählen Sie diesen Eintrag aus.



Der aktualisierte Part erscheint auch im Fenster "Operations".

Zuweisen von Messages zu Operations: (Alternativmethode)

1. Rechtsklicken Sie im Fenster "Operations" auf die Operation **EchoString** und wählen Sie den Befehl **Add Output element**.



Daraufhin wird eine Output: Message an EchoString hinzugefügt.

2. Rechtsklicken Sie im Fenster **Overview** auf den Eintrag **messageName** und drücken Sie F2 (Umbenennen).

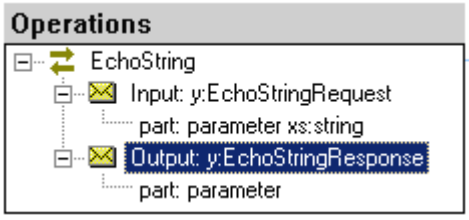
3. Benennen Sie den Message-Eintrag z.B. in "**EchoStringResponse**" um.

4. Rechtsklicken Sie auf den Eintrag und wählen Sie den Befehl **Add message Part (parameter)**.

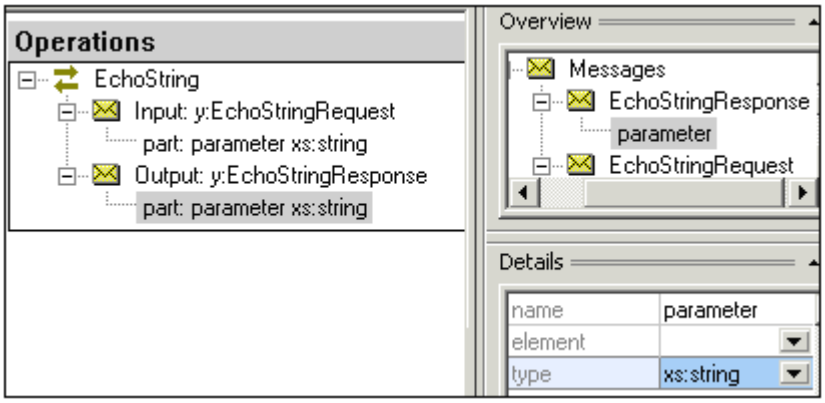
 © 2005 Altova GmbH

5. Klicken Sie auf den Eintrag EchoStringResponse, **ziehen** Sie ihn in das Fenster **Operations** und zwar auf den Eintrag **Output:**.

Neben dem Eintrag Output: erscheint das Erweiterungssymbol. Klicken Sie darauf, um die Ansicht zu erweitern.

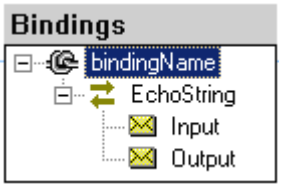


6. Klicken Sie in der **Overview**-Eingabehilfe auf den Eintrag "Parameter" und gehen Sie in der **Details-**Eingabehilfe zur Dropdown-Liste **type**, um **xs:string** einzugeben oder auszuwählen.

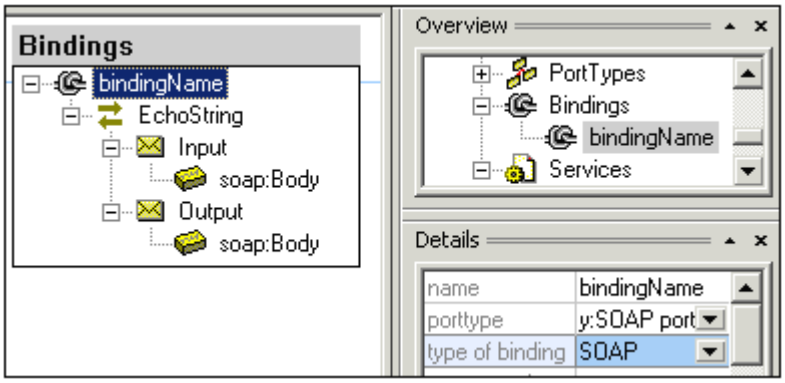


Definieren des Binding-Typs:

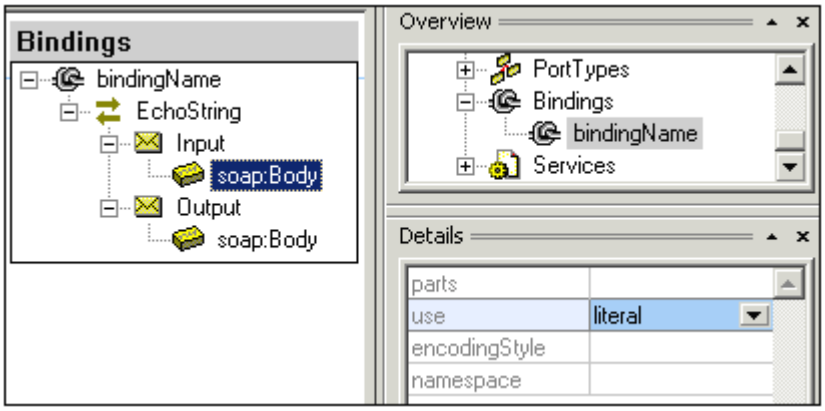
1. Klicken Sie im Fenster "Bindings" auf den Eintrag **bindingName** und anschließend in der **Details-**Eingabehilfe in der Dropdown-Liste **type of binding** auf SOAP.



Daraufhin wird ein Eintrag **soap:Body** zur Input- und Output-Message im Fenster "Bindings" hinzugefügt.



2. Klicken Sie auf den Eintrag **soap:Body** der Input-Message (im Fenster Bindings), klicken Sie in Details-Eingabehilfe auf die Dropdown-Liste **use** und wählen Sie den Eintrag "**literal**". Wiederholen Sie denselben Vorgang bei der Output Message.

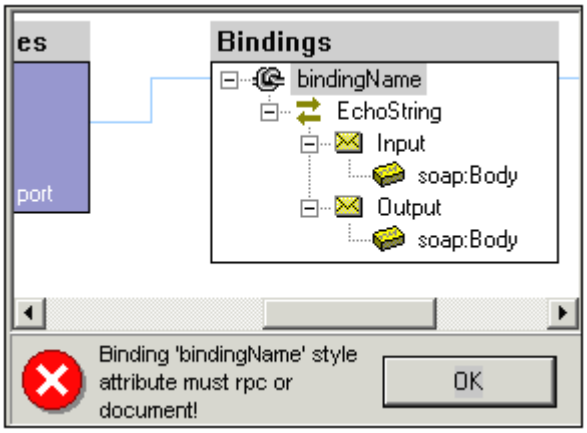


2.3.3 Validieren und Fertigstellung

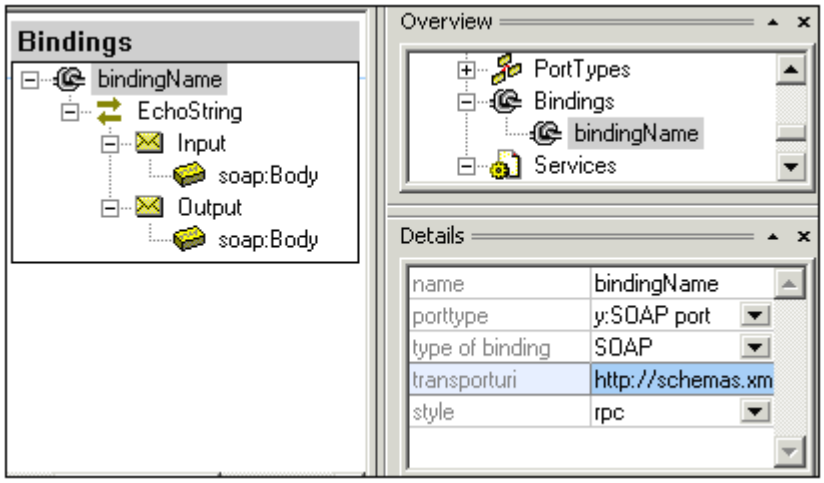
Validieren der WSDL-Datei:
Der graphische WSDL Editor enthält einen integrierten WSDL Validator. Bei Drücken von F8 wird die Datei gegen den W3C WSDL Working Draft validiert.

1. Wählen Sie die Menüoption **XML | validieren** oder drücken Sie die Taste F8. Daraufhin wird eine Meldung angezeigt, dass das bindingName-**Style-Attribut** definiert werden muss.

 © 2005 Altova GmbH

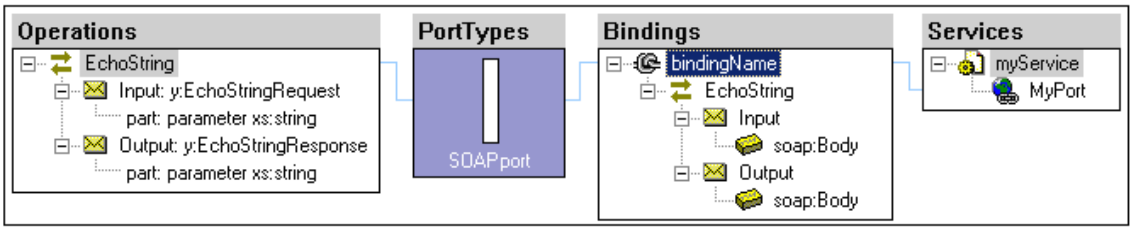


2. Klicken Sie, während der Eintrag **bindingName** aktiv ist, in der **Details-**Eingabehilfe auf die Dropdown-Liste **style** und wählen Sie den Eintrag "**rpc**".
3. Drücken Sie erneut die Taste F8, um die WSDL-Datei zu validieren.
   Es erscheint eine Meldung, dass das **transporturi**-Attribut definiert werden muss.
4. Doppelklicken Sie in der Details-Eingabehilfe auf das Feld **transporturi** und geben Sie die URI, z.B. e.g. **http://schemas.xmlsoap.org/soap/http**, ein.



5. Drücken Sie F8, um die Datei zu validieren.
   Es erscheint die Meldung "Diese Datei ist gültig". Sie könnten die WSDL-Datei nun als Basis für einen Webservice verwenden.

Die fertige WSDL -Datei in der graphischen WSDL-Ansicht:



Die fertige WSDL-Datei in der **Textansicht**:

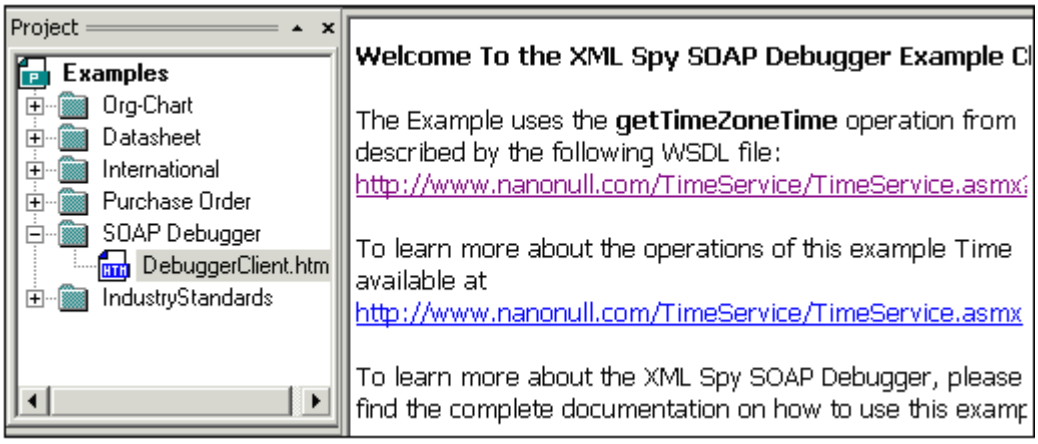
```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:y="http://new.webservice.namespace" targetNamespace="http://new.webservice.namespace">
    <types>
        <xs:schema/>
    </types>
    <message name="EchoStringResponse">
        <part name="parameter" type="xs:string"/>
    </message>
    <message name="EchoStringRequest">
        <part name="parameter" type="xs:string"/>
    </message>
    <portType name="SOAPport">
        <operation name="EchoString">
            <input message="y:EchoStringRequest"/>
            <output message="y:EchoStringResponse"/>
        </operation>
    </portType>
    <binding name="bindingName" type="y:SOAPport">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="EchoString">
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="myService">
        <port name="MyPort" binding="y:bindingName">
            <soap:address location="http://myaddress.com"/>
        </port>
    </service>
</definitions>
```

2.3.4 Herstellen einer Verbindung zu einem Webservice und Kopieren der WSDL-Datei

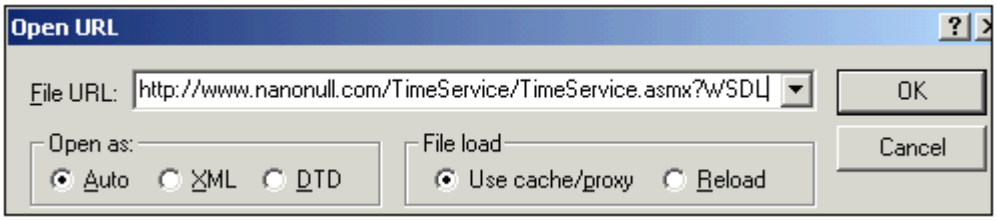
1. Aktivieren Sie das Projekt-Fenster, falls es nicht angezeigt wird, über die Menüoption **Fenster | Projekt-Fenster**.
2. Klicken Sie auf das Erweiterungssymbol neben dem Ordner "SOAP Debugger" und

© 2005 Altova GmbH

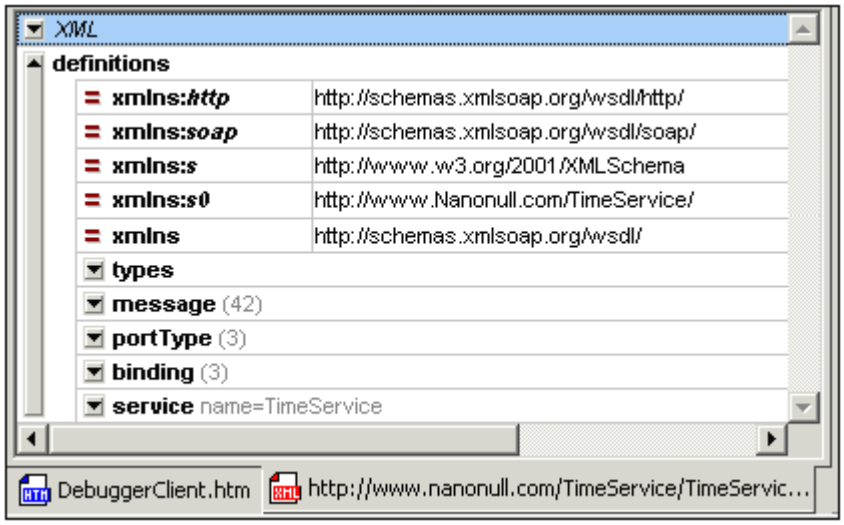
doppelklicken Sie auf die Datei **DebuggerClient.htm**.
Daraufhin wird der SOAP Debugger Example Client im Hauptfenster geöffnet.



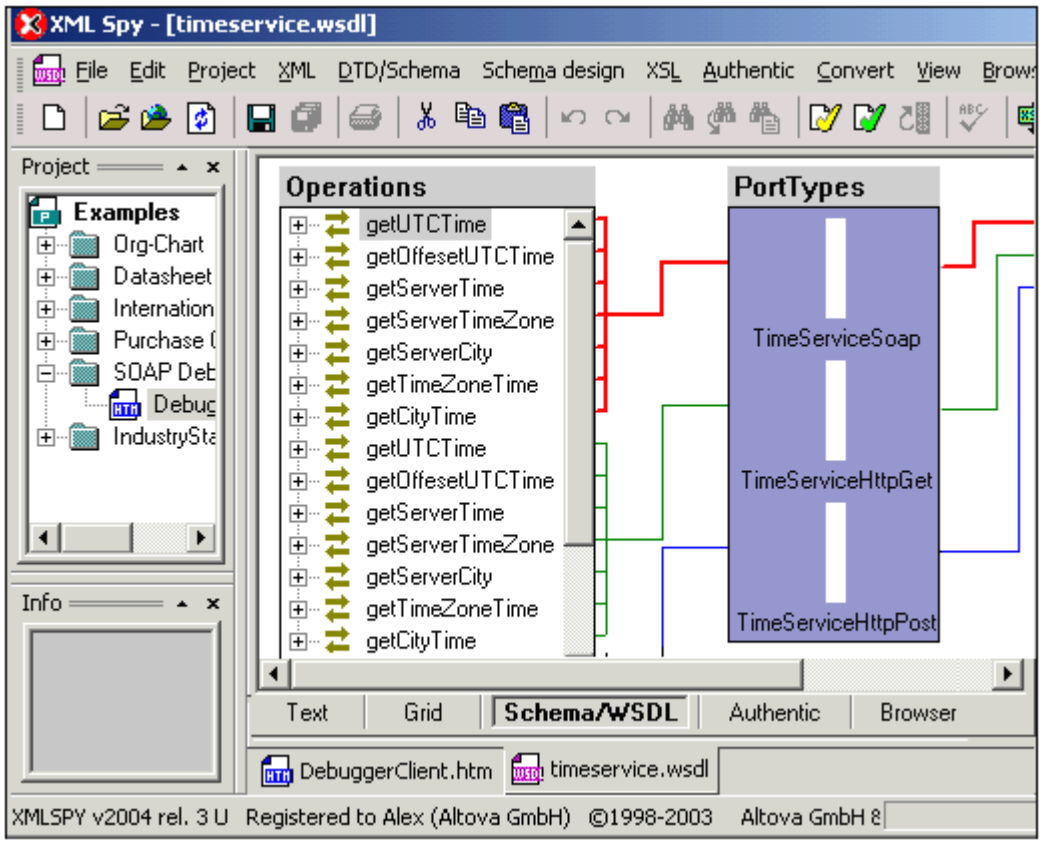
3. Wählen Sie die Menüoption **Datei | URL öffnen...** und geben Sie den Eintrag **http://www.nanonull.com/TimeService/TimeService.asmx?WSDL** im Dialogfeld "URL öffnen" in das Feld "Datei-URL" ein, bzw. kopieren Sie ihn hinein.



4. Klicken Sie auf OK, um die WSDL-Datei zu laden.
Die WSDL-Datei wird in der Enhanced Grid-Ansicht angezeigt (Klicken Sie auf die Definitionsleiste, um mehr von der WSDL-Datei zu sehen).



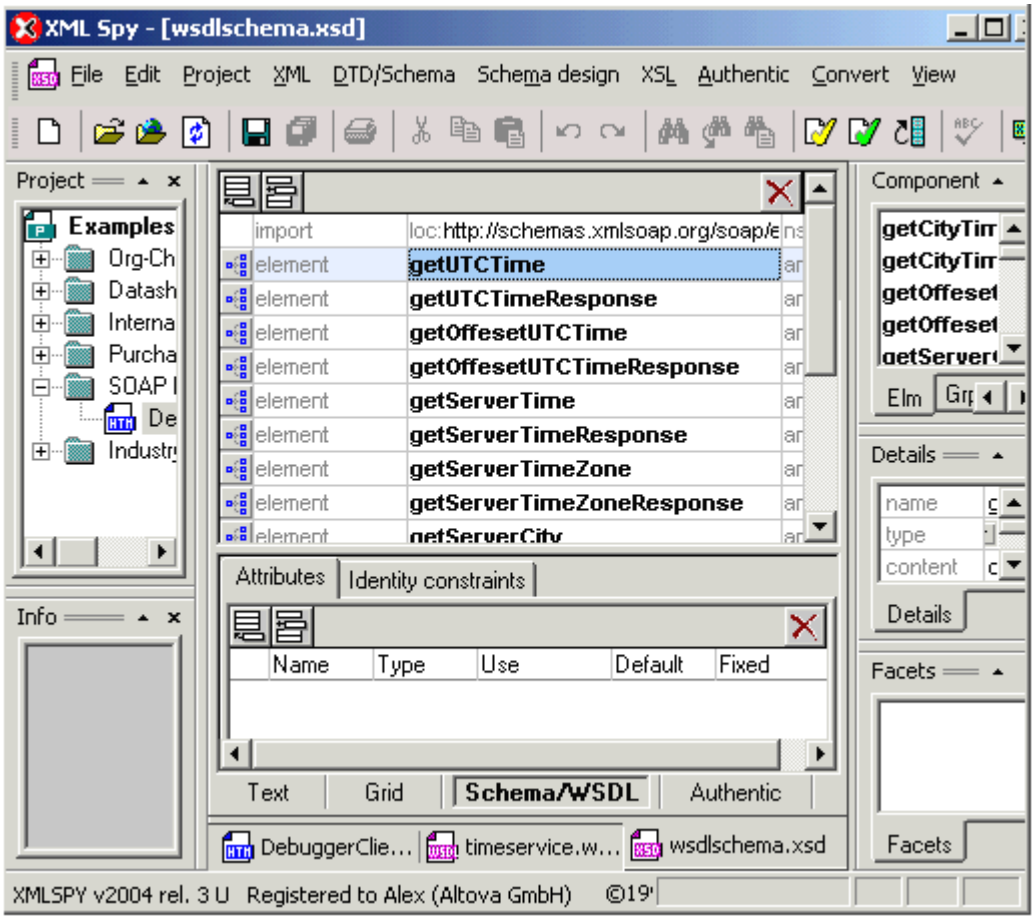
5. Wählen Sie die Menüoption **Datei | Speichern unter...** und speichern Sie die Datei lokal, z.B. unter dem Namen **timeservice.wsdl**.
6. Klicken Sie auf das Symbol Schema/WSDL Design-Ansicht, um die Datei im graphischen WSDL Editor anzusehen.



 © 2005 Altova GmbH

Anzeigen der mit der aktiven WSDL-Datei verknüpften Schema-Datei:
Während im Hauptfenster die Datei timeservice.wsdl aktiv ist:

1. Wählen Sie die Menüoption **WSDL | Types | Schema(s) bearbeiten.**
2. Daraufhin wird im Hauptfenster die Datei **wsdlschema.xsd** geöffnet.



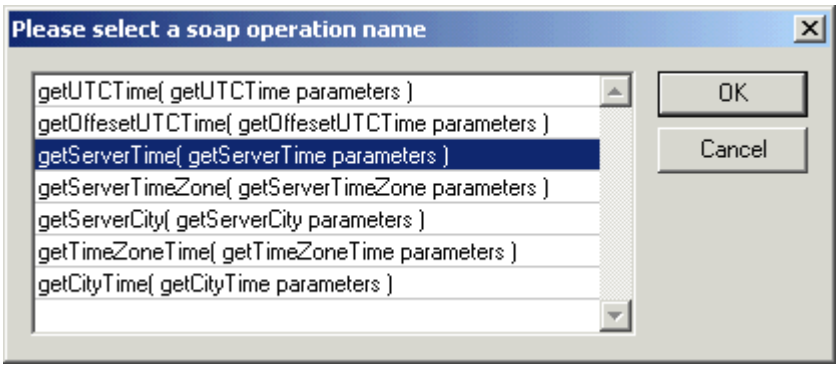
Die Schema-Datei liefert/definiert alle Datentypen, die in der Datei timeservice.wsdl verwendet werden.

Bitte beachten Sie:
Sie können jederzeit auf die dazugehörige WSDL-Datei klicken. Sobald die Datei jedoch aktiv ist, können Sie die Ansicht, in der diese angezeigt wird, also Enhanced Grid-Ansicht oder Textansicht, nicht mehr ändern, da sonst eine Fehlermeldung angezeigt wird. Sie müssen zurück zur Schema-Datei wechseln und diese schließen, bevor Sie die Ansicht der WSDL-Datei wechseln können.

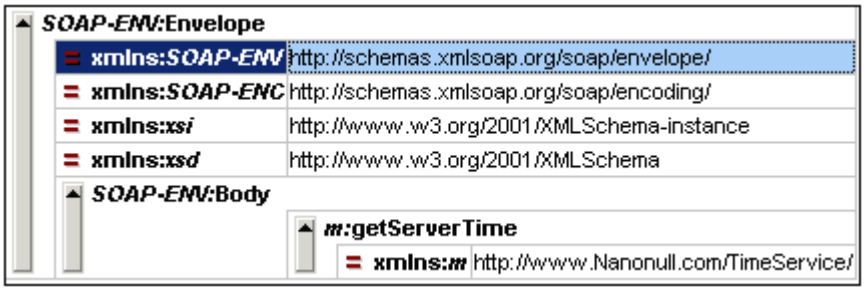
2.3.5 Senden eines SOAP Request von der WSDL-Datei

Senden eines SOAP Request von der WSDL-Datei:
Währen die Datei timeservice.wsdl im Hauptfenster aktiv ist,

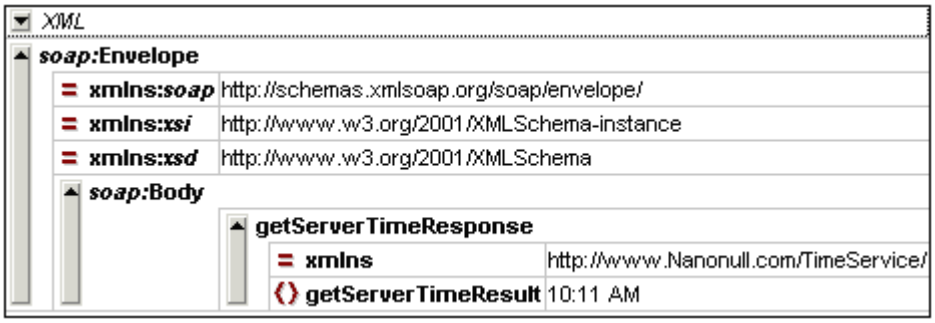
1. Wählen Sie die Menüoption **SOAP | SOAP Request erzeugen.**
2. Klicken Sie auf die Schaltfläche **Fenster...**, um die gerade aktive WSDL-Datei auszuwählen und bestätigen Sie mit OK.
   Wählen Sie im Dialogfeld den Namen einer SOAP Operation aus, z.B. **getServerTime**, und klicken Sie auf OK.



Daraufhin wird ein SOAP Request-Dokument erstellt, das die Operation **getServerTime** enthält.



3. Wählen Sie die Menüoption **SOAP | SOAP Request an Server senden**.
   Das SOAP Response-Dokument wird im Hauptfenster angezeigt und enthält das Element **getServerTimeResult**, das die aktuelle Server-Uhrzeit des Nanonull.com-Zeitservice anzeigt.

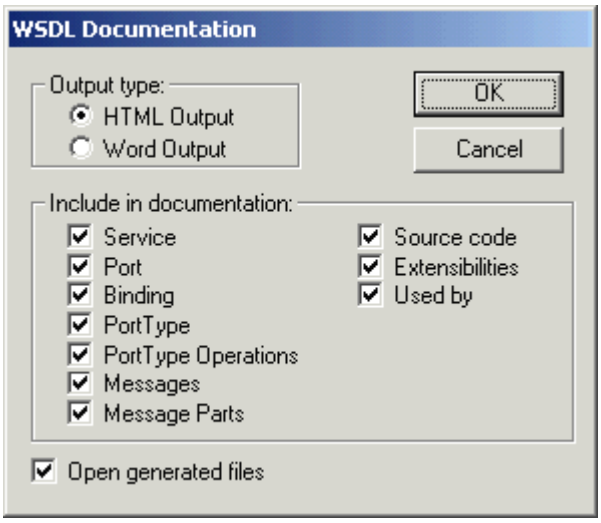


2.3.6 Erstellen von WSDL-Dokumentation

Mit Hilfe der Option **Schema-Dokumentation generieren** können Sie eine detaillierte Dokumentation der aktuellen WSDL-Datei generieren. Sie können die Dokumentation als MS Word- oder HTML-Datei ausgeben und angeben, welche Komponenten inkludiert werden sollen. Miteinander verwandte Elemente werden durch Hyperlinks verbunden, um die Navigation zu vereinfachen.

1. Wählen Sie die Menüoption **WSDL | Schema-Dokumentation generieren**.
   Daraufhin wird das Dialogfeld "WSDL-Dokumentation" geöffnet.
2. Wählen Sie den gewünschten Ausgabetyp: **HTML** oder **Word** und
3. Wählen Sie die spezifischen WSDL-Komponenten, die Sie in der Dokumentation berücksichtigen möchten.

 © 2005 Altova GmbH



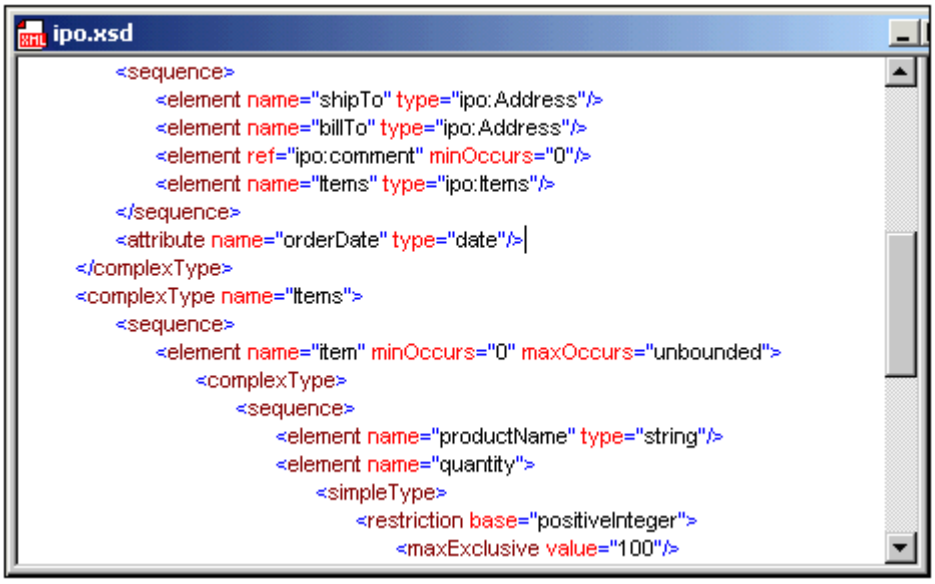
4. Klicken Sie auf OK und geben Sie den Namen der WSDL-Dokumentationsdatei im Dialogfeld "Speichern unter" ein.
   Wenn Sie als Ausgabetyp HTML gewählt haben, wird der WSDL-Dokumentationscode im Hauptfenster angezeigt.

3 Textansicht

In der Textansicht können Sie Ihren Text - Markup und Content - direkt eingeben. Die Textansicht bietet eine Reihe von Funktionen, um Ihr Dokument schnell und korrekt zu erstellen. Zu den wichtigsten Funktionen zählen die folgenden:

- Visuelle Funktionen, zur Verbesserung der Lesbarkeit des Dokuments. Dazu gehören anpassbare Syntaxfärbung (u.a. die Möglichkeit, serverseitigen VBScript- oder JScript-Code in ASP-Seiten zu markieren), Zeilennummerierung, Lesezeichen, ein- und ausklappbare Elemente, Einzüge, anpassbare Schriftarten und Zeilenumbrüche.
- Intelligente Editierfunktionen wie z.B. die Autokomplettierung von Tags und die automatische Eingabe von Attributen und Child-Elementen.
- Kontextsensitive Eingabehilfen, die Ihnen die Elemente, Attribute und Entities, die an der Cursorposition eingefügt werden können, zur Verfügung stellen.
- Drag-and-drop-Funktionen und Funktionen zum Kopieren und Einfügen.

Diese Funktionen werden in den folgenden Kapiteln dieses Abschnitts näher beschrieben.



Um die Textansicht eines Dokuments zu öffnen, klicken Sie auf die Schaltfläche "Text" am unteren Rand des Dokumentenfensters oder wählen Sie **Ansicht | Text**. Die Textansicht dient zum Bearbeiten von Textdateien aller Art, einschließlich nicht-XML-Dokumenten.

© 2005 Altova GmbH

3.1 Visuelle Editierhilfen in der Textansicht

Die Textansicht enthält Funktionen, die die Bearbeitung großer Textpassagen erleichtern: Zeilennummerierung, Lesezeichen, Klappleiste und Einrücklinien. Jede dieser Funktionen kann durch Klicken auf das entsprechende Symbol in der Symbolleiste aktiviert bzw. deaktiviert werden.

Aktiviert/deaktiviert die Zeilennummerierung in allen offenen Dokumenten

Aktiviert/deaktiviert die Klappleiste in allen offenen Dokumenten

Aktiviert/deaktiviert die Lesezeichenleiste

Dient zum Einfügen/Löschen von Lesezeichen

Aktiviert/deaktiviert die Einrücklinien

In der unten gezeigten Abbildung sehen Sie diese Funktionen:

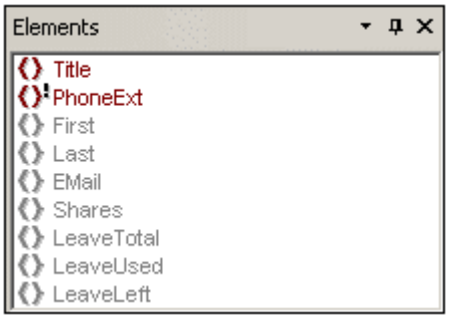
```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!-- edited with XML Spy v4.0 NT beta 1 build Jun 13 2001 (http://www.xmlspy.com) by
3   <schema targetNamespace="http://www.altova.com/IPO" elementFormDefault="unqualit
4     <annotation>
5       <documentation>
6     International Purchase order schema for Example.com
7     Copyright 2000 Example.com. All rights reserved.
8     </documentation>
9     </annotation>
10    <!-- include address constructs -->
11    <include schemaLocation="address.xsd"/>
12    <element name="purchaseOrder" type="ipo:PurchaseOrderType"/>
13    <element name="comment" type="string"/>
14    <complexType name="PurchaseOrderType">
23    <complexType name="Items">
24      <sequence>
44    </complexType>
45    <simpleType name="Sku">
46      <restriction base="string">
47        <pattern value="\d{3}-[A-Z]{2}"/>
```

Beachten Sie die folgenden Funktionen in der oben stehenden Abbildung:

- Die Zeilennummerierung ist aktiviert
- Die Klappleiste ist aktiviert. Mit dem Plus-Zeichen werden eingeklappte (reduzierte) Elemente gekennzeichnet. Mit dem Minus-Zeichen werden ausgeklappte (erweiterte) Elemente gekennzeichnet. Wenn ein Element eingeklappt ist, wird auch die Zeilennummerierung eingeklappt (siehe eingeklappte Zeilen 14 und 24). Wird die Klappleiste deaktiviert, werden alle eingeklappten Elemente automatisch ausgeklappt.
- Die Lesezeichenleiste ist aktiviert. In Zeile 11 wurde ein Lesezeichen gesetzt.
- Die Einrücklinien sind aktiviert. Dabei handelt es sich um vertikale gepunktete Linien, anhand derer Sie besser sehen, wo ein Element beginnt und wo es endet (siehe Zeilen 46 und 47).

3.2 Eingabehilfen in der Textansicht

Element-Eingabehilfe
In der Textansicht werden Elemente, die an der Cursorposition eingefügt werden können, in der Element-Eingabehilfe dunkelrot angezeigt. Obligatorische Elemente haben ein Ausrufezeichen vor dem Elementnamen. Geschwisterelemente von zulässigen Elementen, die selbst an der Cursorposition nicht eingefügt werden können, werden grau angezeigt. Wenn sich die Position des Cursors ändert, ändert sich auch die Liste in der Eingabehilfe und zeigt nur diejenigen Elemente in Rot an, die an dieser Stelle eingefügt werden können, während andere Geschwisterelemente grau angezeigt werden.



Um ein Element an der Cursorposition einzufügen, doppelklicken Sie auf das gewünschte Element. Daraufhin werden der Start- und End-Tag des Elements eingefügt. Auch obligatorische Elemente werden eingefügt, wenn diese Option im Dialogfeld **Optionen** (**Extras | Optionen | Bearbeiten**) aktiviert wurde.

Hinweis: Im Dialogfeld **Optionen** (**Extras | Optionen | Bearbeiten**) können Sie festlegen, dass beim Einfügen eines Elements auch obligatorische Child-Elemente eingefügt werden können.

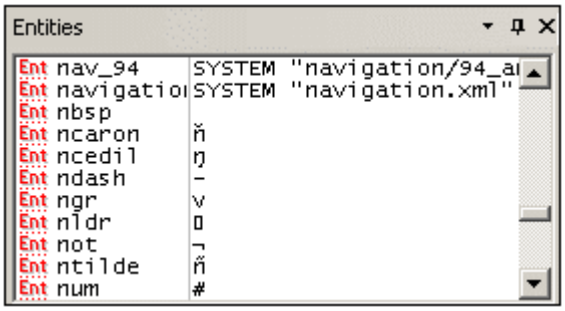
Attribut-Eingabehilfe
Wenn Sie den Cursor in der Textansicht innerhalb des Start-Tags eines Elements und hinter einem Leerzeichen positionieren, werden die für dieses Element deklarierten Attribute angezeigt. Nicht verwendete Attribute werden rot angezeigt, verwendete grau. Obligatorische Attribute weisen ein Ausrufezeichen "**!**" vor dem Namen des Attributs auf.

Um ein Attribut einzufügen, doppelklicken Sie auf das gewünschte Attribut. Das Attribut wird nun an der Cursorposition zusammen mit einem Ist-gleich-Zeichen und Anführungszeichen, innerhalb derer der Attributwert gesetzt wird, angezeigt. Der Cursor wird zwischen die Anführungszeichen gesetzt, sodass Sie den Attributwert direkt eingeben können.

Hinweis: Bestehende Attribute, die kein zweites Mal im aktuellen Element eingefügt werden dürfen, erscheinen grau.

Entities-Eingabehilfe
Alle geparsten oder ungeparsten Entities, die innerhalb einer Zeile (innerhalb des XML-Dokuments) oder in einer externen DTD deklariert sind, werden in der Entities-Eingabehilfe angezeigt.

© 2005 Altova GmbH



Um eine Entity an der Cursorposition einzufügen, doppelklicken Sie auf die gewünschte Entity.

Hinweis: Wenn Sie eine interne Entity einfügen, müssen Sie Ihr Dokument speichern und erneut öffnen, damit die Entity in der Entities-Eingabehilfe angezeigt wird.

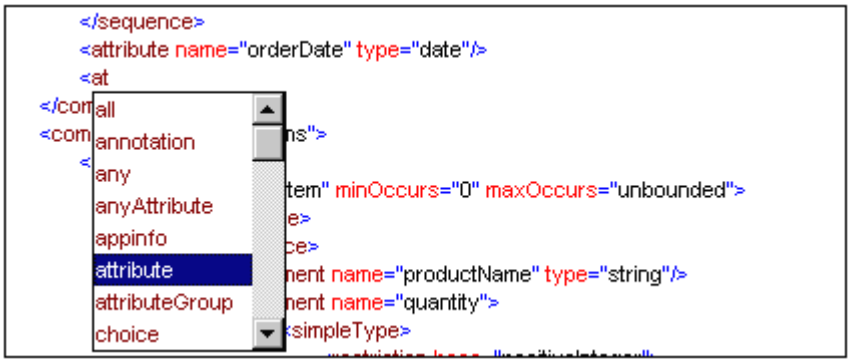
3.3 Bearbeiten von XML-Dokumenten

Syntaxfärbung
Die Syntaxfärbung richtet sich nach der Art des XML Nodes, d.h. je nachdem, ob es sich beim XML Node um ein Element, Attribut, Inhalt, einen CDATA-Abschnitt, Comment oder eine Processing Instruction handelt. Die Texteigenschaften dieser XML Node-Typen können im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Textschriftarten" eingestellt werden.

Start-Tag und End-Tag
Wenn Sie den Cursor innerhalb des Start- oder End-Tags eines XML-Elements setzen, wird der andere Tag des Paars markiert, wenn Sie **Strg+E** drücken. Bei wiederholtem Drücken von **Strg+E** können Sie zwischen dem Start- und End-Tag hin- und herwechseln. Dies ist ein weiteres Hilfsmittel, um die Start- und End-Tags eines XML-Elements leichter zu finden.

Intelligentes Editieren
Wenn Sie mit einem XML-Dokument arbeiten, das auf einer DTD oder einem XML-Schema basiert, stehen Ihnen in XMLSpy 2005 in der Textansicht verschiedene intelligente Editierfunktionen zur Verfügung, mit Hilfe derer Sie schnell das richtige Element, Attribut oder den richtigen Attributwert einfügen können, und zwar dem Content Model entsprechend, das für das Element definiert wurde, das Sie gerade bearbeiten. Intelligentes Editieren funktioniert normalerweise folgendermaßen:

1. Geben Sie **<** (das "kleiner als"-Zeichen) an der Stelle ein, an der Sie ein XML-Element einfügen möchten. Darauf wird eine Popup-Liste angezeigt, die alle Elemente enthält, die an dieser Stelle eingefügt werden dürfen.
2. Geben Sie die ersten paar Zeichen des gewünschten Elements ein. Ein Element in der Popup-Liste, das diese Zeichen enthält, wird markiert.



3. Klicken Sie mit der Maus auf den Eintrag oder drücken Sie die Eingabetaste, um den markierten Eintrag zu übernehmen oder gehen Sie mit Hilfe der Pfeiltasten zum gewünschten Eintrag und klicken Sie darauf oder drücken Sie die Eingabetaste.

Das Popup-Fenster wird auch in den folgenden Fällen angezeigt:

- Wenn Sie die Leertaste drücken, während sich der Cursor zwischen den Tags eines Elements befindet und wenn für dieses Element ein Attribut definiert ist. Das Popup-Fenster enthält alle verfügbaren Attribute.
- Wenn sich der Cursor innerhalb der doppelten Anführungszeichen befindet, die einen Attributwert umschließen, für den eine Aufzählung mit Werten zur Auswahl steht. Im Popup-Fenster werden die aufgezählten Werte angezeigt.
- Wenn Sie das Zeichen **</** (den Beginn eines End-Tags) eingeben, wird im Popup-Fenster der Name des zu schließenden Elements angezeigt.

© 2005 Altova GmbH

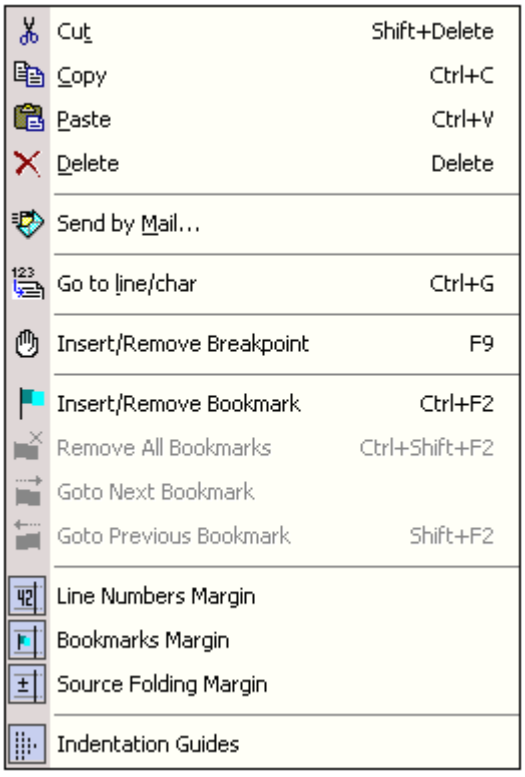
Autokomplettierung
Wenn Sie XML-Dokumente in der Textansicht bearbeiten, wird oft die Wohlgeformtheit eines Dokuments verletzt, so können z.B. unter Umständen End-Tags fehlen, falsch geschrieben sein oder in der Struktur an einer falschen Stelle stehen.
XMLSpy 2005 ergänzt den Start- und End-Tag von Elementen automatisch und fügt alle obligatorischen Attribute ein, sobald Sie auf der Tastatur den Elementnamen fertig eingegeben haben. Außerdem wird der Cursor automatisch zwischen den Start- und End-Tag des Elements gesetzt, sodass Sie gleich weitere Child-Elemente oder Inhalt eingeben können:

```
<img src="" alt="">|</img>
```

Mit dem Befehl Wohlgeformtheit prüfen können Sie jederzeit überprüfen, ob das Dokument wohlgeformt ist. Diese Überprüfung wird auch automatisch jedes Mal beim Öffnen oder Speichern eines Dokuments durchgeführt.

Drag-and-Drop und Kontextmenüs
Sie können einen Textblock auch mit Drag-and-Drop an eine andere Stelle verschieben und häufig verwendete Editier-Befehle (wie Ausschneiden, Kopieren, Einfügen, Löschen, Als Mail senden und Gehe zu Zeile/Zeichen) mit einem Rechtsklick direkt über ein Kontextmenü aufrufen.

Die anderen Befehle im Kontextmenü ermöglichen Ihnen Lesezeichen zu setzen/löschen usw. und die Textansicht anzupassen.



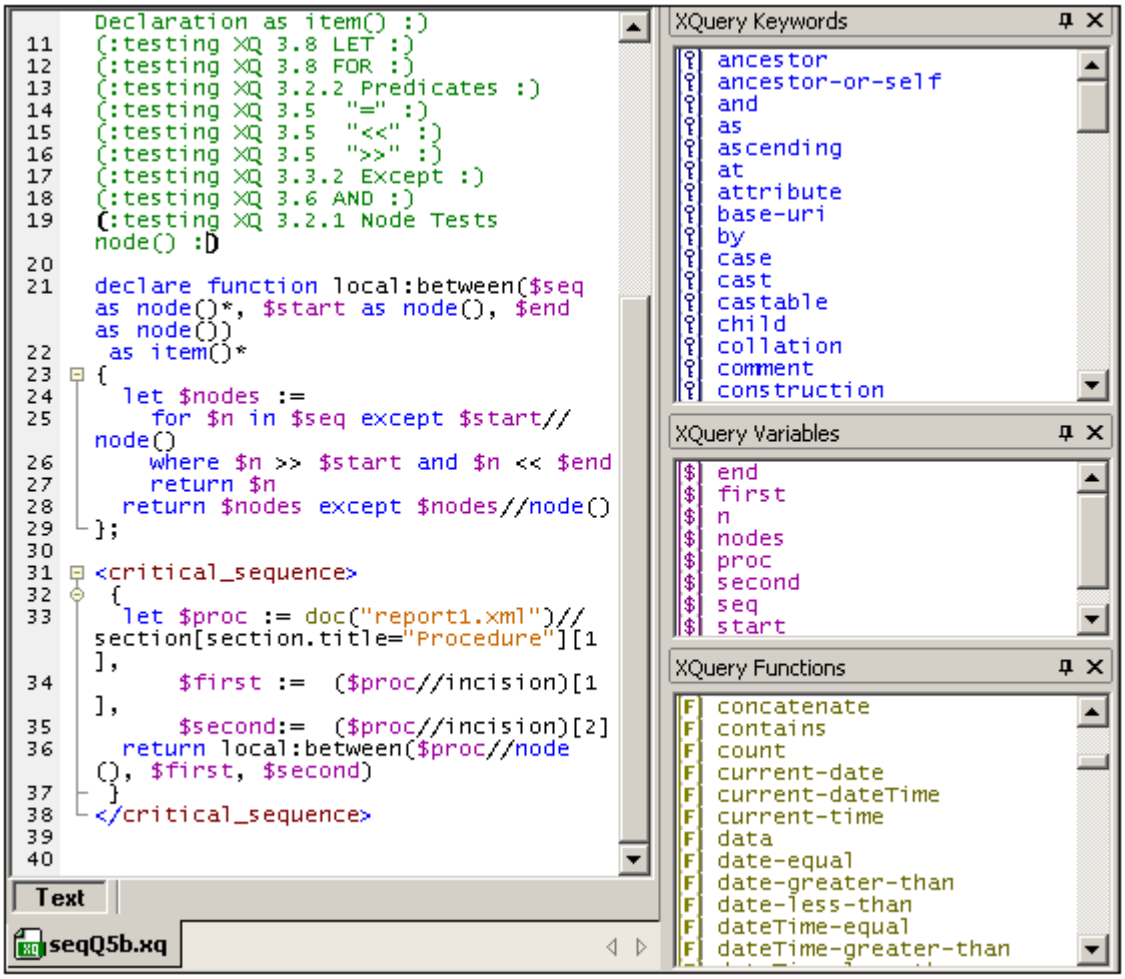
Suchen und Ersetzen
Mit Hilfe der Befehle Suchen und Ersetzen können Sie Text schnell finden und ändern. Diese Befehle unterstützen auch Regular Expressions, sodass Ihnen damit leistungsstarke Suchfunktionen zur Verfügung stehen. (Nähere Informationen siehe Bearbeiten | Suchen.)

Unbeschränktes Rückgängigmachen
XMLSpy 2005 bietet Ihnen für alle Editiervorgänge unbeschränktes Rückgängigmachen und Wiederherstellen.

© 2005 Altova GmbH

3.4 Bearbeiten von XQuery-Dokumenten

In der Textansicht können Sie XQuery-Dokumente bearbeiten. Die Eingabehilfen, Syntaxfärbung und das intelligente Editieren unterscheiden sich dabei von denen, die für die Textansicht von XML-Dokumenten zur Verfügung stehen (*siehe Abbildung unten, Zeilennummerierung und Klappleisten stehen nur in der Enterprise und der Professional Edition zur Verfügung)*. Dieser Textansichtsmodus wird XQuery-Modus genannt. Zusätzlich dazu können Sie Ihr XQuery-Dokument in der Textansicht validieren und den Code in einem XQuery-Dokument (mit einer optionalen XML-Datei, wenn nötig) mit Hilfe des Altova XQuery-Prozessors ausführen .



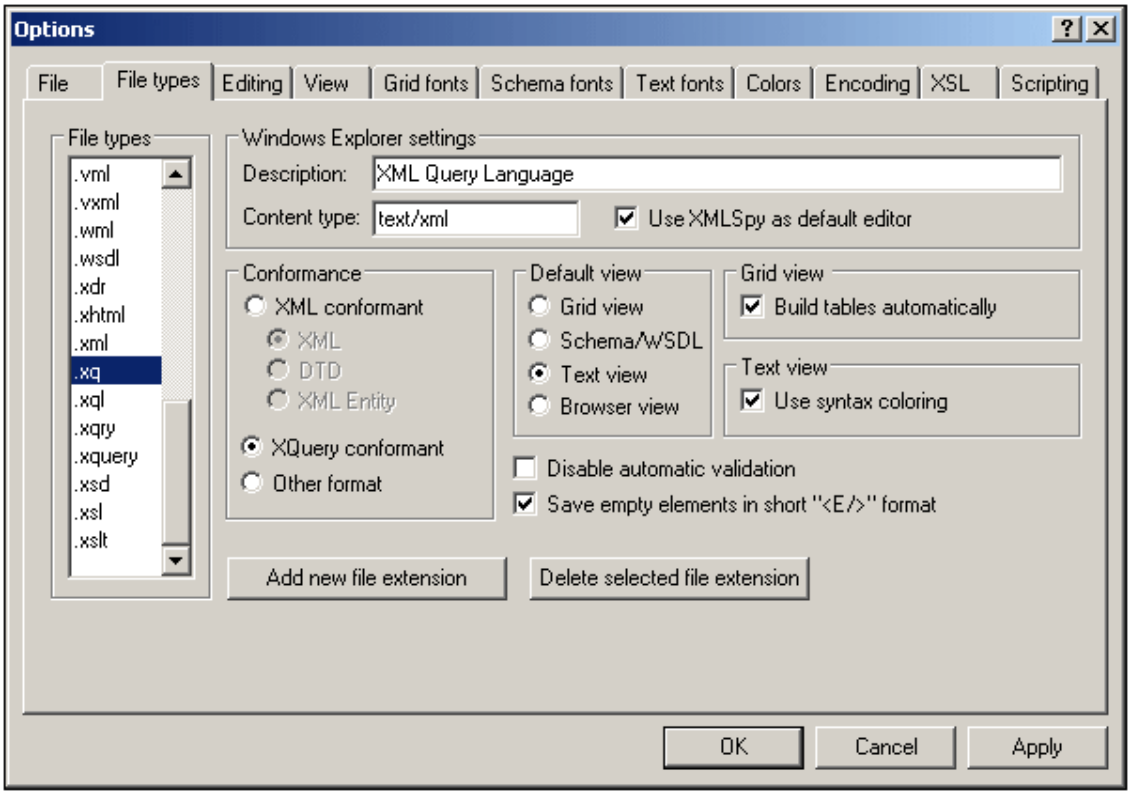
Hinweis: XQuery-Dateien können nur in der Textansicht bearbeitet werden. Für XQuery-Dateien stehen keine anderen Ansichten zur Verfügung.

Nähere Informationen dazu, wie der Altova-XQuery-Prozessor implementiert ist und XQuery-Dateien verarbeitet, finden Sie unter "Implementierung des XQuery-Prozessors".

3.4.1 Öffnen eines XQuery-Dokuments

Ein XQuery-Dokument wird automatisch im XQuery-Modus der Textansicht geöffnet, wenn es XQuery-konform ist. Dateien mit der Dateierweiterung `.xq`, `.xql`, und `.xquery` sind in XMLSpy 2005 als XQuery-konform vordefiniert. Auf dem Register "Dateiarten" des Dialogfelds "Optionen" (**Extras | Optionen**) (siehe Abb. unten) können Sie weitere Dateierweiterungen als

XQuery-konform definieren. Fügen Sie dazu die Dateierweiterung zur Liste der Dateiarten hinzu und definieren Sie sie als XQuery-konform.



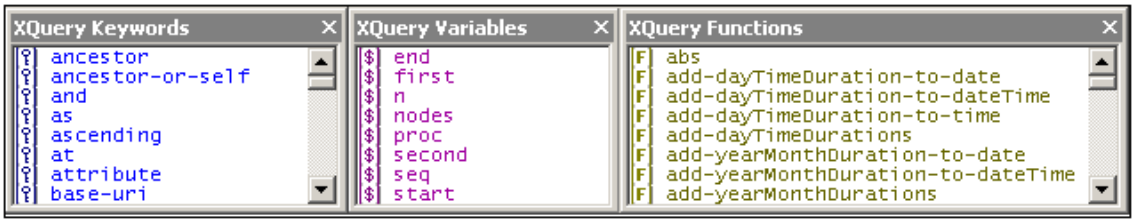
Außerdem sollten Sie die folgenden Windows Explorer-Einstellungen vornehmen:

- Beschreibung: XML Query Language
- Inhaltstyp: Text/xml

Wenn Sie XMLSpy 2005 als Standardeditor für XQuery-Dateien verwenden wollen, aktivieren Sie das entsprechende Kontrollkästchen.

3.4.2 XQuery Eingabehilfen

In der Textansicht stehen Ihnen Im XQuery-Modus drei Eingabehilfen zur Verfügung: XQuery-Schlüsselwörter (blau), XQuery-Variablen (violett) und XQuery-Funktionen (olivgrün).



Beachten Sie bitte die folgenden Punkte:

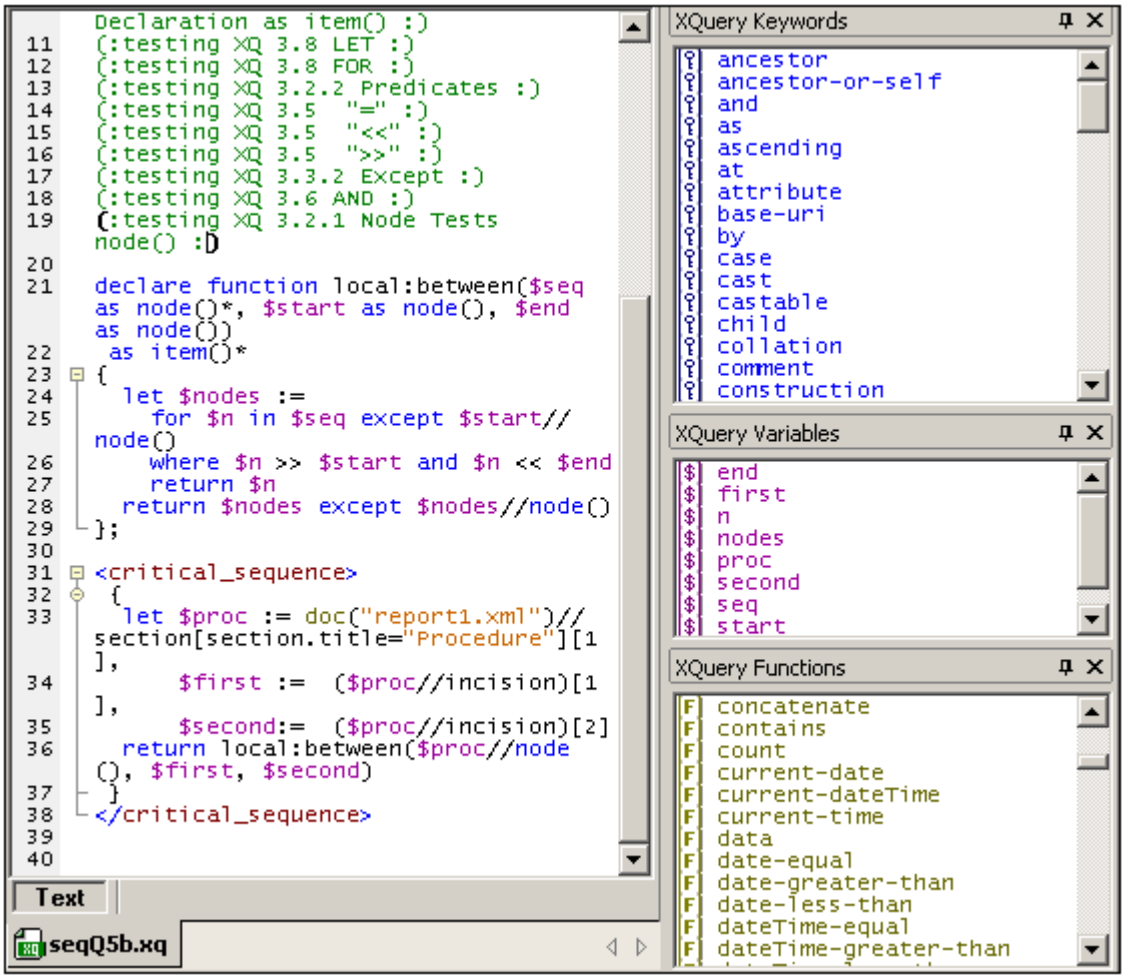
- Die Farbe der Einträge in den drei Eingabehilfen ist unterschiedlich und entspricht der im Text verwendeten Syntaxfärbung. Diese Farben können nicht geändert werden.
- Die aufgelisteten Schlüsselwörter und Funktionen sind diejenigen, die vom Altova XQuery-Prozessor unterstützt werden.

- Die Variablen werden im XQuery-Dokument selbst definiert. Wenn Sie in der Textansicht ein $-Zeichen und ein weiteres Zeichen eingeben, erscheint das Zeichen in der Variablen-Eingabehilfe (es sei denn, es existiert eine Variable, die aus genau diesem Zeichen besteht). Sobald ein eingegebener Variablenname mit einem bereits existierenden Variablennamen übereinstimmt, verschwindet der neu eingegebene Variablenname aus der Eingabehilfe.
- Zum Navigieren in einer Eingabehilfe klicken Sie auf einen Eintrag in der Eingabehilfe und bewegen Sie sich mittels Bildlaufleiste, Mausrad oder der Bild auf- bzw. Bild ab-Taste durch die Liste.

Um einen der in den Eingabehilfen aufgelisteten Einträge in das Dokument einzufügen, setzen Sie den Cursor an die gewünschte Stelle und doppelklicken Sie auf den Eintrag. In XQuery stellen einige Zeichenfolgen sowohl ein Schlüsselwort als auch eine Funktion dar (`empty`, `unordered` und `except`). Diese Strings werden immer als Schlüsselwörter eingegeben (blau) - selbst wenn Sie die Funktion dieses Namens in der Funktionen-Eingabehilfe auswählen. Wenn eine Funktion blau angezeigt wird, ist sie durch die Klammern, die auf den Funktionsnamen folgen, als solche zu erkennen.

3.4.3 XQuery Syntaxfärbung

Ein XQuery-Dokument kann sowohl aus XQuery-Code als auch XML-Code bestehen. In diesem Abschnitt wird die Standard-Syntaxfärbung für den XQuery-Code beschrieben. Die Syntaxfärbung für XML-Code in einem XQuery-Dokument ist dieselbe wie die in normalen XML-Dokumenten verwendete. Alle Syntaxfärbungseinstellungen (sowohl für XQuery-Code als auch für XML-Code) werden im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Textschriftarten" vorgenommen. Beachten Sie, dass XML-Elemente XQuery-Code enthalten können. Der XQuery-Code wird dabei in geschwungene Klammern **{}** gesetzt (*Beispiel siehe Abbildung; Zeilennummerierung und Klappleiste nur in der Enterprise und der Professional Edition*).



Im XQuery-Modus der Textansicht wird für XQuery-Code die folgende Standard-Syntaxfärbung verwendet:

- **(: Comments einschließlich 'Smiley'-Trennzeichen sind grün :)**
- XQuery-Schlüsselwörter sind blau: **Schlüsselwort**
- XQuery-Variablen, einschließlich des Dollar-Zeichens sind violett: **$start**
- XQuery-Funktionen, **nicht** aber die einschließenden Klammern sind olivgrün: **Funktion()**
- Strings sind orange: **"Prozedur"**
- Der gesamte andere Text, wie z.B. Path-Ausdrücke ist schwarz (*wie der unterstrichene Text unterhalb*). Dies sieht so aus:

```
for $s in doc("report1.xml")//section[section.title =
"Procedure"]
return ($s//incision)[2]/instrument
```

Sie können diese Standardfarben sowie andere Schriftarteinstellungen im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Textschriftarten" ändern.

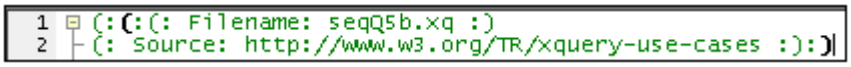
Hinweis: Im oben gezeigten Screenshot wird ein farbiges Klammerpaar für einen Comment schwarz und fett angezeigt. Dies geschieht aufgrund der Funktion zum Anzeigen von Kammerpaaren (siehe Intelligentes Editieren von XQuery).

 © 2005 Altova GmbH

3.4.4 Intelligentes Editieren von XQuery

Im XQuery-Modus der Textansicht stehen Ihnen die folgenden intelligenten Editierfeatures zur Verfügung.

Anzeigen von Klammerpaaren
Die Funktion zum Anzeigen von Klammerpaaren markiert die öffnende und schließende Klammer eines Klammerpaars, sodass der Inhalt eines Klammerpaars klar ersichtlich ist. Dies ist v.a. bei verschachtelten Klammern, wie dies bei XQuery-Comments (*siehe Abbildung unten*) der Fall ist, hilfreich.

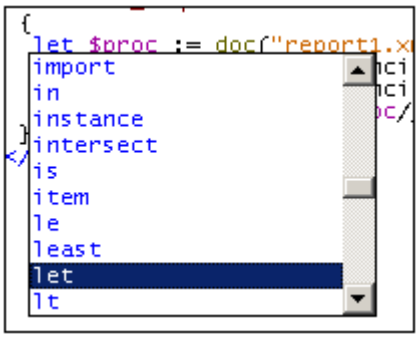


Die Klammerpaaranzeige wird aktiviert, indem Sie den Cursor entweder unmittelbar vor oder unmittelbar hinter eine (öffnende oder schließende) Klammer setzen. Diese Klammer wird zusammen mit ihrem Gegenstück (schwarz und fett) hervorgehoben. Beachten Sie die Position des Cursors im obigen Screenshot.

Die Klammerpaaranzeige ist für runde **( )**, eckige **[ ]** und geschwungene Klammern **{ }** aktiv. Eine Ausnahme bilden spitze Klammern **<>**, die für XML-Tags verwendet werden.

Hinweis: Wenn Sie den Cursor innerhalb eines Start- oder End-Tags eines XML-Elements setzen, wird der andere Teil des Klammerpaars markiert, wenn Sie **Strg+E** drücken. Bei wiederholtem Drücken von **Strg+E** können Sie zwischen dem Start- und End-Tag hin- und herwechseln. Dies ist ein weiteres Hilfsmittel, um Start- und End-Tag eines XML-Elements leichter zu finden.

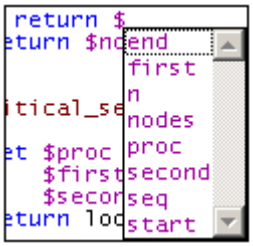
Schlüsselwörter
XQuery-Schlüsselwörter sind in Query-Ausdrücken verwendete Anweisungen und werden blau angezeigt. Wenn Sie den Cursor in oder unmittelbar vor oder hinter ein Schlüsselwort platzieren, wird das Schlüsselwort ausgewählt. Wenn Sie nun **Strg+Leertaste** drücken, wird in einem Popup-Menü eine komplette Liste aller Schlüsselwörter angezeigt. Sie können durch die Liste scrollen und auf ein Schlüsselwort doppelklicken, um das ausgewählte Schlüsselwort zu ersetzen.



In der oben gezeigten Abbildung wurde der Cursor in das Schlüsselwort `let` gesetzt. Wenn Sie auf ein Schlüsselwort aus der Liste doppelklicken, wird das Schlüsselwort `let` durch dieses ersetzt.

Variablen
Variablennamen ist das **$**-Zeichen vorangestellt. Variablen werden violett angezeigt. Die Vorgangsweise bei dieser intelligenten Editierfunktion ist ähnlich wie bei der für Schlüsselwörter. Es gibt zwei Arten, um die Popup-Liste aller Variablen in einem Dokument aufzurufen:

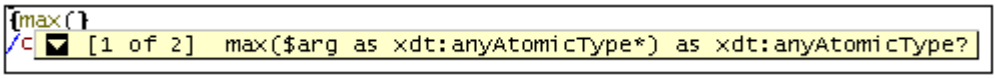
- Geben Sie zuerst ein **$**-Zeichen ein und drücken Sie **Strg+Leertaste**
- Wählen Sie eine Variable aus und drücken Sie **Strg+Leertaste**. (Zum Auswählen einer Variable setzen Sie den Cursor unmittelbar hinter das **$** -Zeichen, in den Namen einer Variable oder unmittelbar hinter den Namen der Variable.)



Um eine Variable (während der Eingabe) hinter dem **$**-Zeichen einzufügen oder eine ausgewählte Variable zu ersetzen, doppelklicken Sie im Popup-Menü auf die gewünschte Variable.

Funktionen
Genau wie bei Schlüsselwörtern und Variablen wird hier ein Popup-Menü mit vordefinierten Funktionen angezeigt, wenn Sie eine Funktion (angezeigt in Grün) auswählen und **Strg+Leerzeichen** drücken. (Zum Auswählen einer Funktion setzen Sie den Cursor in einen Funktionsnamen oder unmittelbar vor oder hinter den Namen. Der Cursor darf nicht zwischen die Klammern gesetzt werden, die auf den Funktionsnamen folgen). Wenn Sie im Popup-Menü auf einen Funktionsnamen doppelklicken, wird der ausgewählte Funktionsname durch die Funktion aus dem Popup-Menü ersetzt.

Um einen Tipp anzuzeigen, der die Bezeichnung der eingefügten Funktion enthält (*siehe Abbildung unten)*, setzen Sie den Cursor unmittelbar hinter die öffnende Klammer und drücken Sie **Strg+Leertaste.**



Der nach unten weisende Pfeil bedeutet, dass es mehrere Funktionen mit demselben Namen gibt. (Solche Funktionen unterscheiden sich hinsichtlich der Art und Anzahl der Argumente oder der Art des Returns, den sie haben). Wenn Sie auf den Tipp klicken, wird die nächste Funktion angezeigt und wenn Sie mehrmals hintereinander auf den Tipp klicken, werden der Reihe nach alle Funktionen dieses Namens angezeigt. Alternativ dazu können Sie auch durch Drücken von **Strg+Umschalt+Nach unten** oder **Strg+Umschalt+nach oben** durch die Funktionen wechseln.

Visuelle Orientierungshilfen
Die Textein- und -ausklappfunktion ist bei geschwungenen XQuery-Klammern, bei XQuery Comments, XML-Elementen und XML-Comments aktiv.

3.4.5 Validierung und Ausführung von XQuery-Dokumenten

Validierung von XQuery-Dokumenten
Um ein XQuery-Dokument zu validieren, gehen Sie vor wie folgt:

1. Machen Sie das XQuery-Dokument zum aktiven Dokument.
2. Wählen Sie **XML | Validieren** oder drücken Sie **F8** oder klicken Sie in der Symbolleiste auf das Symbol .

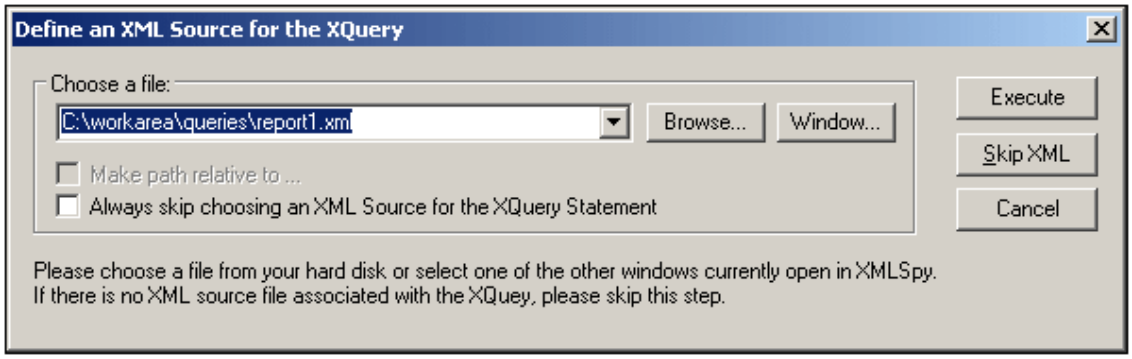
 © 2005 Altova GmbH

Das Dokument wird daraufhin auf die korrekte XQuery-Syntax überprüft.

Ausführen von XQuery-Dokumenten
XQuery-Dokumente werden in XMLSpy 2005 mit dem integrierten XQuery 1.0-Prozessor ausgeführt. Das Resultat wird in einem Fenster in XMLSpy 2005 angezeigt.

Normalerweise ist ein XQuery-Dokument nicht mit einem einzelnen XML-Dokument verknüpft, da XQuery-Ausdrücke mit Hilfe der `doc( )`-Funktion eine beliebige Anzahl von XML-Dokumenten auswählen können. In XMLSpy 2005 können Sie jedoch vor dem Ausführen eines einzelnen XQuery-Dokuments ein XML-Quelldokument dafür auswählen. In diesem Fall ist der Dokument-Node des ausgewählten XML-Quelldokuments der Context-Node, der auf der Root-Ebene des XQuery-Dokument als Ausgangspunkt angegegeben ist. Pfade, die mit einem Schrägstrich beginnen, werden mit diesem Dokument-Node als Context-Element aufgelöst. Um ein XQuery-Dokument auszuführen, gehen Sie vor wie folgt:

1. Machen Sie das XQuery-Dokument zum aktiven Dokument.
2. Wählen Sie **XSL/XQuery | XQuery Transformation** oder klicken Sie in der Symbolleiste auf das Symbol . Daraufhin erscheint das Dialogfeld "Definieren Sie eine XML-Quelle für die XQuery".



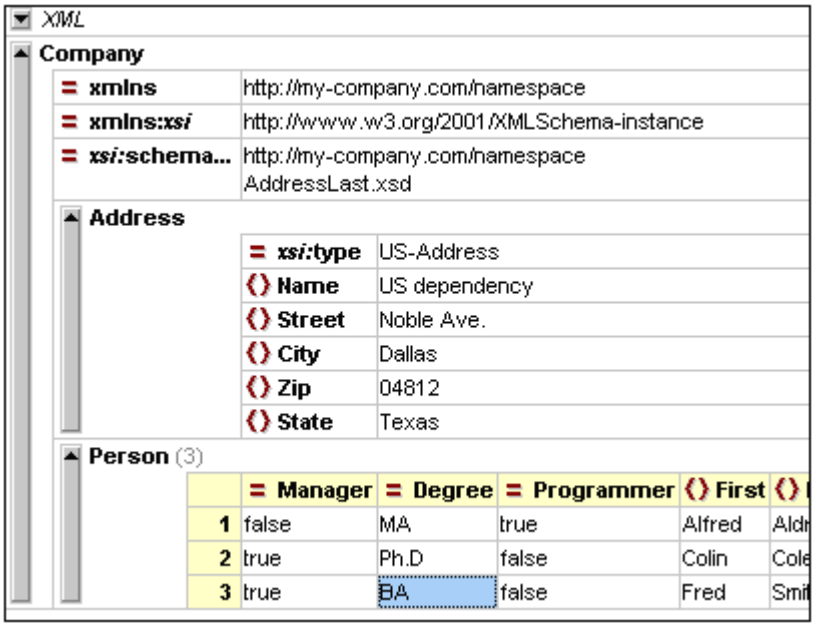
3. Sie können nun ein XML-Quelldokument auswählen, wenn Sie dessen Dokument-Node als Context-Node für die Root-Ebene des XQuery-Dokuments zuweisen wollen. Andernfalls können Sie dieses Dialogfeld überspringen. Zum Auswählen der XML-Datei verwenden Sie entweder die Schaltfläche "Durchsuchen" oder die Schaltfläche "Fenster" (damit werden Dateien, die in XMLSpy 2005 geöffnet sind und jene die sich in XMLSpy-Projekten befinden, aufgelistet).

Das Ergebnisdokument wird als temporäre Datei generiert, die im gewünschten Dateiformat und mit der gewünschten Erweiterung in einem beliebigen Ordner gespeichert werden kann.

Nähere Informationen darüber, wie der Altova XQuery-Prozessor implementiert ist und XQuery-Dateien verarbeitet, finden Sie unter Implementierung des XQuery-Prozessors.

4 Enhanced Grid-Ansicht

Durch Klicken auf die Schaltfläche **Grid** am unteren Rand des Dokumentenfensters wird die Enhanced Grid-Ansicht (oder Grid-Ansicht) geöffnet. Sie zeigt die hierarchische Struktur von XML-Dokumenten mittels verschachtelter Datenbehälter an. Diese können für eine bessere Übersicht über die gesamte Dokumentstruktur erweitert oder reduziert werden. In der Grid-Ansicht können Sie Inhalt und Struktur eines Dokuments einfach bearbeiten.



Ein hierarchisch dargestelltes Objekt (z.B. XML-Deklaration, Dokumenttyp-Deklaration oder ein Element, das Child-Elemente enthält), wird durch eine graue senkrechte Leiste auf der linken Seite dargestellt, die einen kleinen nach oben weisenden Pfeil enthält. Durch Klicken auf die Leiste können Sie das Datenelement erweitern oder reduzieren. Ein Element wird durch das Symbol {} gekennzeichnet, ein Attribut durch das Symbol =.

Anzeige und Navigation

- Der Inhalt eines hierarchischen Elements hängt von dessen Eigenschaften ab. Bei Elementen handelt es sich dabei z.B. normalerweise um Attribute, Character Data, Comments und Child-Elemente. Attribute werden immer aufgelistet und erscheinen in der Reihenfolge der Input-Datei. Den Attributen folgen alle weiteren Objekte genau in der Reihenfolge, in der diese in der Quelldatei angegeben wurden. Sie können beliebig mittels Drag & Drop neu arrangiert werden. Diese Änderung wird auch in den Quelldaten übernommen.
- Falls ein Element nur Character Data enthält, werden diese Daten in der Zeile angezeigt, in der sich auch das Element befindet. Dieses Element wird dann nicht als hierarchisch angesehen. Character Data für andere Elemente werden eingerückt (zusammen mit Attributen und eventuellen Child-Elementen) und als "Text" gekennzeichnet.
- Bei reduzierten Elementen werden ihre Attribute und deren Werte in derselben Zeile in Grau angezeigt. Diese Attribut-Vorschau ist besonders hilfreich beim Editieren von XML-Dokumenten mit einer großen Anzahl von Elementen desselben Namens, die sich nur durch Inhalt oder Attribute unterscheiden (z.B. Datenbank-Applikationen).
- Mit Hilfe der Pfeiltasten können Sie die Laufleiste in der Grid-Ansicht bewegen.

© 2005 Altova GmbH

- Mit Hilfe der + und – Tasten der Zahlentastatur können Sie Datenelemente erweitern und reduzieren.

Anpassen der Grid-Ansicht

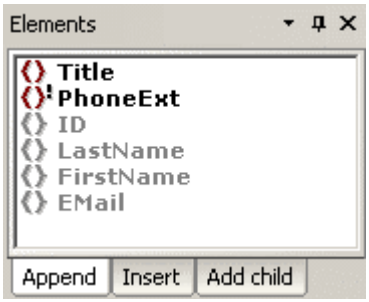
- Um die Größe von Spalten anzupassen, platzieren Sie den Cursor über den entsprechenden Rahmen und ziehen Sie diesen an die gewünschte Stelle.
- Um die Größe einer Spalte an die Breite des breitesten Eintrags anzupassen, doppelklicken Sie auf den rechten Rand der gewünschten Spalte.
- Um die Spaltenbreite so anzupassen, dass der gesamte Inhalt zu sehen ist, wählen Sie den Menüeintrag Ansicht | Optimale Breite oder klicken Sie auf das Symbol "Optimale Breite" .
- Die Höhe der Zellen ist vom jeweiligen Inhalt abhängig. Sie kann über die Menüoption **Extras | Optionen | Ansicht | Enhanced Grid-Ansicht** "Max. Zellenhöhe xxx Zeilen" angepasst werden.

Hinweis: Wenn Sie Daten in der Grid-Ansicht markieren und in die Textansicht wechseln, werden diese auch in der Textansicht markiert angezeigt.

Intelligentes Editieren
Die Grid-Ansicht gestattet intelligentes Editieren, wenn das XML-Dokument auf einem Schema basiert. Nähere Informationen dazu finden Sie unter Editieren in der Grid-Ansicht. Die in der Grid-Ansicht verfügbaren Eingabehilfen sind im Folgenden beschrieben.

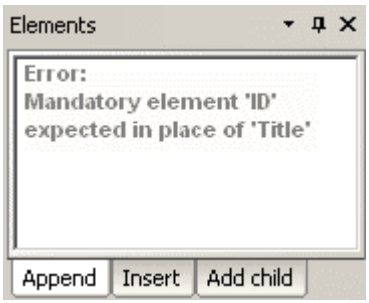
Eingebettete Datenbank/Tabellen-Ansicht
In der Grid-Ansicht können sich wiederholende Elemente in einer Datenbank/Tabellen-Ansicht angezeigt werden. Diese Funktion kann zum Editieren von jeder beliebigen XML, XSD, XSLT usw. Datei verwendet werden. Die in der Datenbank/Tabellen-Ansicht verwendeten Eingabehilfen sind weiter unten beschrieben.

Element-Eingabehilfe
In der Grid-Ansicht hat die Element-Eingabehilfe drei Register: Anhängen, Einfügen und Child hinzufügen. Auf dem Register **Anhängen** werden Elemente angezeigt, die nach allen Geschwisterelementen des aktuellen Elements angehängt werden können; auf dem Register **Einfügen** werden alle Elemente angezeigt, die vor dem aktuellen Element eingefügt werden können und auf dem Register **Child hinzufügen** werden alle jene Elemente angezeigt, die als Child-Element des aktuellen Elements eingefügt werden können.



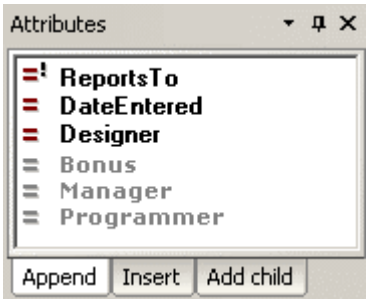
Um ein Element einzufügen, wählen Sie das entsprechende Register und doppelklicken Sie auf das gewünschte Element. Obligatorische Elemente werden durch ein vorangestelltes Ausrufezeichen gekennzeichnet. Geschwisterelemente von zulässigen Elementen, die selbst an der Cursorposition nicht eingefügt werden können, sind ausgegraut und deaktiviert.

Wenn Sie eine Struktur erstellen, die nicht dem in Ihrem Schema definierten Content Model entspricht, zeigt der integrierte Validierungsparser im Element-Eingabehilfefenster der Grid-Ansicht eine Fehlermeldung an.



Hinweis: Im Dialogfeld **Optionen** (**Extras | Optionen | Bearbeiten**) können Sie festlegen, dass obligatorische Child-Elemente eingefügt werden können, wenn ein Element eingefügt wird.

Attribut-Eingabehilfe
Die Attribut-Eingabehilfe enthält eine Liste aller für das derzeit bearbeitete Element verfügbaren Attribute. Obligatorische Attribute sind durch ein Ausrufezeichen "**!**" vor dem Namen des Attributs gekennzeichnet. Wurde für dieses Element bereits ein Attribut angegeben, wird dieses Attribut grau angezeigt.



- Um die Attribute auf den Registern "Anhängen" und "Einfügen" zu verwenden, wählen Sie in der Grid-Ansicht ein existierendes Attribut oder ein Element, das ein Child-Element des Parent-Elements des Attributs ist, aus und doppelklicken Sie in der Eingabehilfe auf das gewünschte Attribut.
- Um die Attribute aus dem Register "Child hinzufügen" zu verwenden, wählen Sie in der Grid-Ansicht das Parent-Element des Attributs und doppelklicken Sie in der Eingabehilfe auf das gewünschte Attribut.

Hinweis: Existierende Attribute, die bereits hinzugefügt wurden und zu diesem Element kein zweites Mal hinzugefügt werden dürfen, werden grau angezeigt.

Entities-Eingabehilfe
In der Entities-Eingabehilfe werden alle geparsten oder nicht geparsten Entities angezeigt, die innerhalb der Zeile oder in einer externen DTD deklariert sind. Wenn in der Grid-Ansicht ein Text- oder Attribut-Node ausgewählt ist, ist das Register "Child hinzufügen" leer, da solche Nodes laut Definition keine Child-Elemente enthalten können.

Um in der Grid-Ansicht eine Entity einzufügen, platzieren Sie den Cursor an der gewünschten Position im Textfeld oder wählen Sie das gewünschte Feld aus; wählen Sie anschließend das entsprechende Register aus und doppelklicken Sie auf die Entity. Beachten Sie die folgenden Regeln:

- Wenn der Cursor in ein Textfeld (z.B. auch ein Attributwertfeld) gesetzt wird, wird die Entity an der Cursorposition eingefügt.
- Ist ein Element mit einem nur-Text-Child-Element (d.h. `#PCDATA` oder `simpleType`) ausgewählt und der Cursor befindet sich nicht im Textfeld, wird vorhandener Text durch die Entity **ersetzt.**

 © 2005 Altova GmbH

- Wenn ein nicht-Text-Feld ausgewählt ist, wird die Entity an einer Stelle als Text erstellt, die dem ausgewählten Register der Eingabehilfe entspricht.

Hinweis: Wenn Sie eine interne Entity hinzufügen, müssen Sie Ihr Dokument speichern und erneut öffnen, damit die Entity in der Entities-Eingabehilfe angezeigt wird.

4.1 Editieren in der Grid-Ansicht

Beim Editieren eines schemabasierten (DTD oder XML-Schema) XML-Dokuments in der Enhanced Grid-Ansicht stehen Ihnen auf Basis von aus dem Schema stammenden Informationen intelligente Editierfeatures zur Verfügung. Diese Features sind im Folgenden aufgelistet.

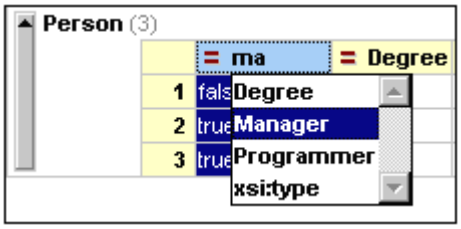
Einfügen oder Anhängen von Element- oder Attributnamen
Zum Einfügen oder Anhängen eines Attributs oder Elements relativ zum ausgewählten Datenelement, können Sie entweder die Befehle im XML-Menü oder die Symbole in der **Attribut- und Element-Symbolleiste** verwenden. Falls diese Symbolleiste nicht sichtbar ist, wählen Sie die Menüoption **Extras | Anpassen** und aktivieren Sie auf dem Register **Symbolleisten** den Eintrag **Attr & Elem**. Eine ausführliche Beschreibung zur Verwendung der Befehle der Symbolleiste und des XML-Menüs finden Sie im Abschnitt Menü "XML".

Um ein Element oder Attribut einzufügen oder anzuhängen, gehen Sie folgendermaßen vor:

1. Wählen Sie das Datenelement, relativ zu welchem das Element oder Attribut eingefügt oder angehängt werden soll.
2. Klicken Sie auf das entsprechende Symbol in der Element- und Attributsymbolleiste oder wählen Sie den Befehl im XML-Menü aus. Daraufhin wird im Raster ein neuer Eintrag erstellt und es erscheint ein Popup-Fenster mit verfügbaren Element- oder Attributoptionen.
3 Wählen Sie im Popup-Fenster das gewünschte Element oder Attribut aus und bestätigen Sie die Auswahl durch Klicken oder Drücken der Eingabetaste. Alternativ dazu können Sie den Namen des gewünschten Elements oder Attributs auch eingeben.

Sie werden feststellen, dass die verschiedenen Eingabehilfen je nachdem, was in der Grid-Ansicht gerade ausgewählt ist, ständig aktualisiert werden. Im Info-Fenster werden jeweils wichtige Informationen zum ausgewählten Element/Attribut angezeigt.

Bearbeiten eines Element- oder Attributnamens
Wenn Sie den Namen eines Elements oder Attributs bearbeiten, wird ein Popup-Menü mit den verfügbaren Optionen geöffnet. Diese Optionen sind abhängig von der Position des Elements und dem im Schema definierten Content Model. Ein ähnliches Popup-Menü wird angezeigt, wenn der Inhalt eines Elements oder Attributs durch eine Enumeration oder Choice eingeschränkt wird.

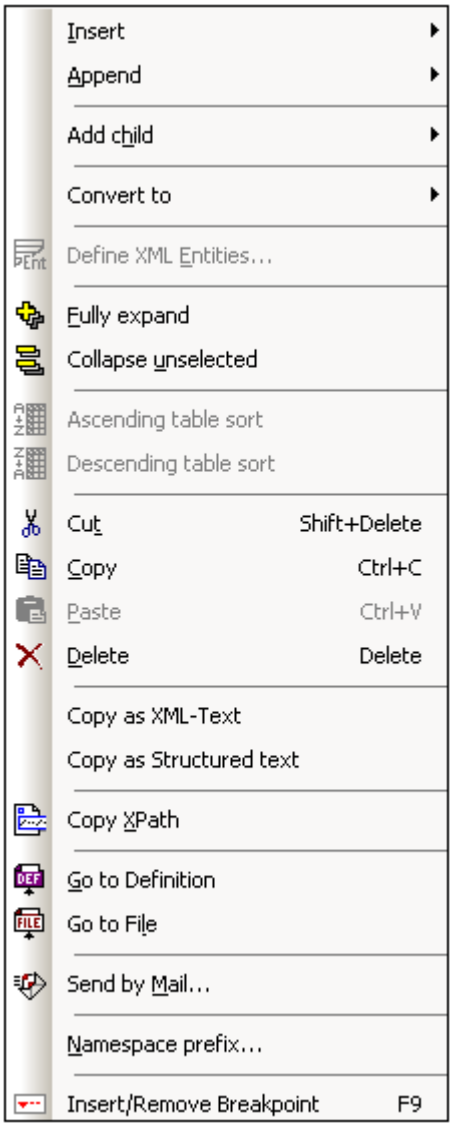


So bearbeiten Sie einen Element- oder Attributnamen:

1. Doppelklicken Sie auf den Element- oder Attributnamen. Es erscheint ein Popup-Fenster mit den verfügbaren Optionen.
2. Wählen Sie das gewünschte Element oder Attribut aus dem Popup-Menü aus.
3. Bestätigen Sie die Auswahl durch Drücken der **Eingabetaste** oder Klicken auf den Namen. (mit **Esc** verwerfen Sie die Änderungen.)

 © 2005 Altova GmbH

Kontextmenü der Grid-Ansicht
Zusätzlich zu den Befehlen, die über die verschiedenen Menüs und Symbolleisten zur Verfügung stehen, bietet die Grid-Ansicht auch ein Kontextmenü (siehe unten), in dem die am häufigsten verwendeten Befehle aus den verschiedensten Menüs zusammengefasst sind. Klicken Sie mit der rechten Maustaste auf das Datenelement, für das Sie eine Aktion durchführen wollen, um das Kontextmenü zu öffnen.



4.2 Datenbank/Tabellen-Ansicht

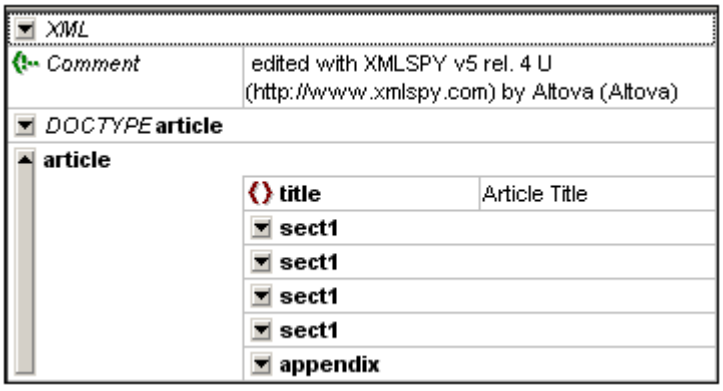
Die **Datenbank/Tabellen-Ansicht** (im Folgenden Tabellenansicht genannt) ist in die Enhanced Grid-Ansicht integriert. Sich wiederholende Elemente werden in dieser Ansicht in einem Tabellenformat angezeigt. Die Tabellenansicht unterscheidet sich von der normalen Grid-Ansicht insofern, als darin für jeden Child-Typ des in Tabellenform dargestellten Elements eine Spalte angelegt wird. Sie können anschließend für ganze Spalten oder Auswahlbereiche Eigenschaften ändern.

Sehen Sie sich als Beispiel das folgende XML-Dokument an:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- edited with XMLSPY v5 rel. 4 U (ht
3  <!DOCTYPE article SYSTEM "C:\Program Fi
4  <article>
5    <title>Article Title</title>
6    <sect1>
7      <title>Section 1</title>
8      <simpara/>
9    </sect1>
10   <sect1>
11     <title>Section 2</title>
12     <para/>
13   </sect1>
14   <sect1>
15     <title>Section 3</title>
16     <para/>
17     <para/>
18   </sect1>
19   <sect1>
20     <title>Section 4</title>
21     <para/>
22   </sect1>
23   <appendix>
24     <title/>
25     <para/>
26   </appendix>
27 </article>
```

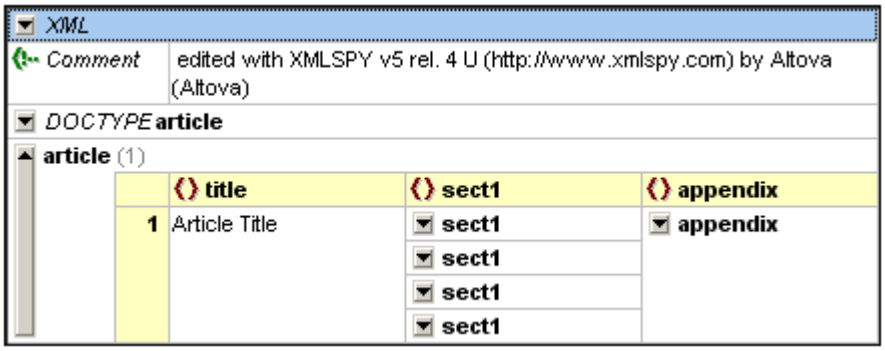
Das Dokument-Element ist `article` und `article` hat die folgende Sequenz von Child-Elementen: ein `title`-Element, vier `sect1`-Elemente und ein `appendix`-Element. Jedes `sect1`- und `appendix`-Element hat ein `title`-Element gefolgt von einer Reihe von `para` oder `simpara`-Elementen.

In der normalen Grid-Ansicht wird dieses Dokument folgendermaßen dargestellt:

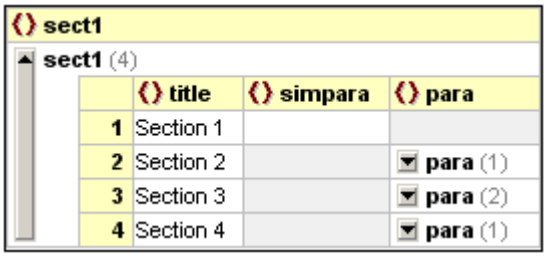


Hier sehen Sie nun die Tabellenansicht dieses Dokuments - besser gesagt des Elements `article`. (Um zu dieser Ansicht zu gelangen, wählen Sie das Element `article` in der

normalen Grid-Ansicht aus (durch Anklicken) und klicken Sie auf das Symbol **Als Tabelle anzeigen** . Alternativ dazu wählen Sie den Menübefehl **XML | Tabelle | Als Tabelle anzeigen (F12)**.)



Beachten Sie, dass jedem Child-Element von `article` (dem als Tabelle angezeigten Element) - also `title`, `sect1` und `appendix` - eine Spalte zugewiesen wurde und dass jede Instanz jedes Child-Typs in der entsprechenden Spalte aufgelistet ist. Beachten Sie außerdem, dass die Tabellenstruktur nur bis zur Ebene der Child-Elemente geht (die `sect1`-Elemente selbst werden nicht als Tabelle dargestellt). Um die `sect1`-Elemente als Tabelle darzustellen, wählen Sie eines der `sect1`-Elemente in der Spalte `sect1` aus und klicken Sie auf . Die `sect1`-Elemente werden nun in Tabellenform dargestellt (siehe Abbildung unten): Jedem der Child-Elemente wird eine Spalte in der `sect1`-Tabelle zugewiesen.



Wenn es für eine der vier Instanzen von `sect1` ein Child-Element gibt (`title`, `simpara` oder `para`), so hat die entsprechende Zelle einen weißen Hintergrund (z.B. `simpara` in der ersten Instanz von `sect1`). Wenn es für eine Instanz kein Child-Element gibt, so hat die entsprechende Zelle einen grauen Hintergrund (z.B. `para` in der ersten Instanz von `sect1`). Daraus ersieht man, dass für jeden unterschiedlichen Child-Typ aller `sect1`-Instanzen eine eigene Spalte angelegt wurde.

Hinweis:

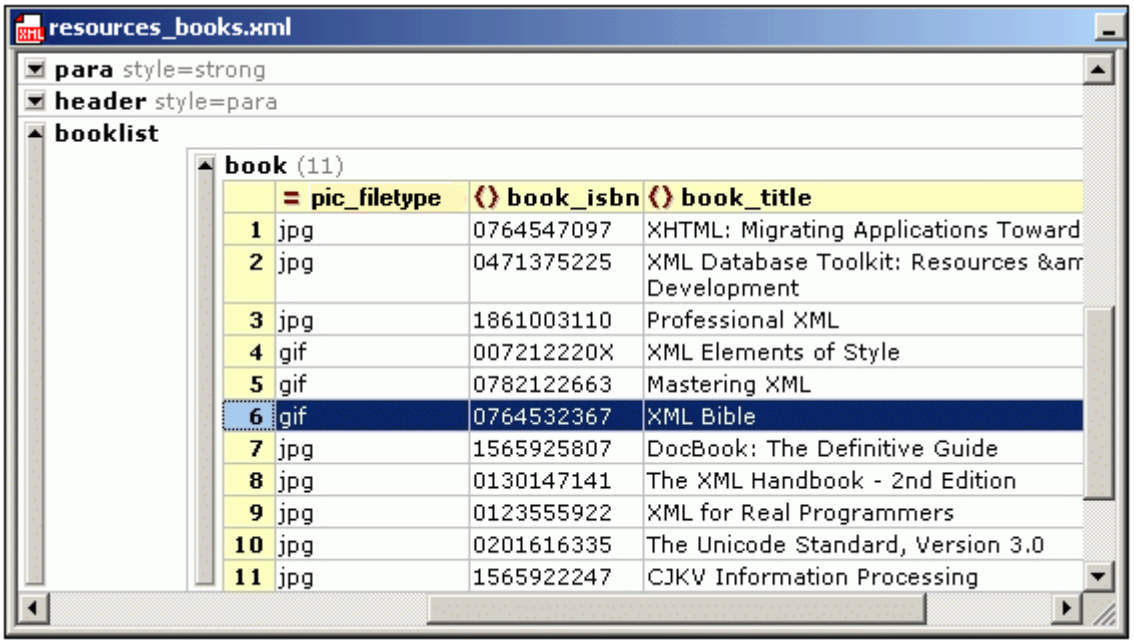
- Auch Attribute werden als Child Nodes betrachtet. Auch für Attribute werden Spalten erstellt.
- Sie können zwischen der normalen Grid-Ansicht und der Tabellenansicht wechseln, indem Sie das gewünschte Element auswählen und auf klicken oder **F12** drücken.
- Wenn Sie das Dokument in der Tabellenansicht ansehen und in die Textansicht wechseln, wird es beim Zurückwechseln in die Grid-Ansicht in der normalen Grid-Ansicht angezeigt.

Bearbeitung von Tabellendaten
Sie können Tabellendaten auf folgende Art bearbeiten:

- Spalten mit Drag-and-Drop der Spaltenüberschrift verschieben.
- Spaltendaten für Text-Node mit Hilfe des Menübefehls **XML | Tabelle | Aufsteigend sortieren** (auch absteigend sortieren) sortieren.

- Zeilen mit Hilfe des Menübefehls **XML | Tabelle | Zeile einfügen** (auch Zeile anhängen) einfügen (oder anhängen).

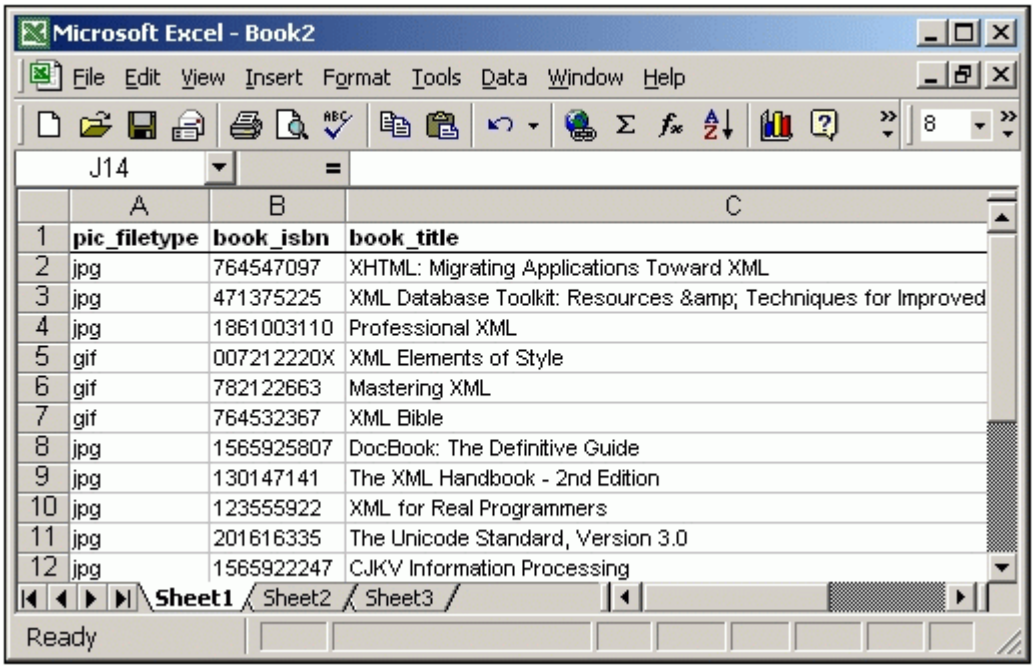
Verschieben von Daten zwischen der Tabellenansicht und externen Applikationen
Die Tabellenstruktur der Tabellenansicht eignet sich auch dazu, Daten aus einem Tabellenkalkulationsprogramm zu übernehmen und umgekehrt. Um Daten aus der Tabellenansicht in eine andere Applikation zu kopieren, wählen Sie die gewünschten Nodes in der Tabelle aus und kopieren Sie die Daten mit Hilfe des Befehls Als strukturierten Text kopieren direkt in z.B. ein Excel Sheet. (Sie können Nodes in der Tabellenansicht durch Klicken auf die Zelle selbst, auf die Spalten- oder Zeilenüberschrift (siehe unten) oder die gesamte Tabelle auswählen. Wenn Sie die gesamte Tabelle oder Spaltenüberschriften auswählen, wird auch der Text der Spaltenüberschriften ausgewählt; ansonsten nicht)



	= pic_filetype	{} book_isbn	{} book_title
1	jpg	0764547097	XHTML: Migrating Applications Toward
2	jpg	0471375225	XML Database Toolkit: Resources &am Development
3	jpg	1861003110	Professional XML
4	gif	007212220X	XML Elements of Style
5	gif	0782122663	Mastering XML
6	gif	0764532367	XML Bible
7	jpg	1565925807	DocBook: The Definitive Guide
8	jpg	0130147141	The XML Handbook - 2nd Edition
9	jpg	0123555922	XML for Real Programmers
10	jpg	0201616335	The Unicode Standard, Version 3.0
11	jpg	1565922247	CJKV Information Processing

Im oben gezeigten Screenshot sehen Sie 11 Buch-Elemente die in der Tabellenansicht in Tabellenform angezeigt werden. Um die gesamte Tabelle in ein Excel Sheet zu kopieren, müssten Sie das Element `book(11)` auswählen, es als strukturierten Text kopieren und in ein Excel Sheet einfügen. Wenn Sie Zelle A1 auswählen und die Tabelle einfügen, sieht das Ergebnis aus, wie unten gezeigt.

© 2005 Altova GmbH



	A	B	C
1	**pic_filetype**	**book_isbn**	**book_title**
2	jpg	764547097	XHTML: Migrating Applications Toward XML
3	jpg	471375225	XML Database Toolkit: Resources & Techniques for Improved
4	jpg	1861003110	Professional XML
5	gif	007212220X	XML Elements of Style
6	gif	782122663	Mastering XML
7	gif	764532367	XML Bible
8	jpg	1565925807	DocBook: The Definitive Guide
9	jpg	130147141	The XML Handbook - 2nd Edition
10	jpg	123555922	XML for Real Programmers
11	jpg	201616335	The Unicode Standard, Version 3.0
12	jpg	1565922247	CJKV Information Processing

Der Datenaustausch funktioniert in beide Richtungen. Sie können daher auch Daten aus jedem Tabellenkalkulationsprogramm kopieren und in der Tabellenansicht direkt in eine Tabelle einfügen. Gehen Sie dabei folgendermaßen vor:

1. Wählen Sie in der externen Applikation einen Bereich aus und kopieren Sie ihn (in die Zwischenablage - in Windows-Systemen mit **Strg+C**).
2. Wählen Sie in Ihrem XML-Dokument in der Tabellenansicht eine einzelne Zelle aus.
3. Fügen Sie die kopierten Daten mit **Strg+V** ein.

Die Daten werden in XMLSpy 2005 in die Tabelle eingefügt. Die Tabellenstruktur der Originaltabelle bleibt dabei erhalten und die Tabelle wird beginnend in der Zelle eingefügt, die in der Tabellenansicht ausgewählt ist. Beachten Sie bitte Folgendes:

- Wenn diese Zellen in der Tabellenansicht bereits Daten enthalten, werden die alten Daten durch die neuen überschrieben.
- Wenn für die neuen Daten mehr Zellen benötigt werden, werden diese erstellt.
- Werden jedoch mehr Spalten für die neuen Daten benötigt, werden diese **nicht** erstellt.
- Die Daten in den Zellen werden Inhalt der Elemente, für die die entsprechenden Zellen stehen.

Für komplexere Datenaustauschaufgaben bietet XMLSpy 2005 außerdem einige einzigartige Konvertierungsfunktionen zum direkten Importieren oder Exportieren von XML-Daten aus jeder beliebigen Textdatei, jedem Word-Dokument oder jeder Datenbank.

5 Schema/WSDL Design-Ansicht

Die **Schema/WSDL Design-Ansicht** ist bei XML-Schema- und WSDL-Dokumenten aktiv. Die Ansicht ändert sich, je nachdem, ob ein XML-Schema oder ein WSDL-Dokument aktiv ist und wird Schema Design-Ansicht bzw. WSDL Design-Ansicht genannt. Diese Ansichten bieten eine grafische Darstellung des XML-Schema- bzw. WSDL-Dokuments und gestatten Ihnen, XML-Schemas und WSLD-Dokumente auf intuitive Art zu erstellen und zu bearbeiten. Wenn Sie eine lizenzierte Version von SchemaAgent auf Ihrem Rechner installiert haben, haben Sie beim Erstellen von Schemas nicht nur Zugriff auf die Komponenten des derzeit in XMLSpy 2005 aktiven Schemas, sondern auch auf die Komponenten vieler anderer Schemas, die in anderen Repositories auf Ihrem Netzwerk gespeichert sind. Im Abschnitt Arbeiten mit SchemaAgent wird beschrieben, wie Sie SchemaAgent installieren und damit arbeiten.

Diese Ansichten sind in den Unterkapiteln dieses Abschnitts ausführlich beschrieben. Beispiele, wie man XML-Schemas und WSDL-Dokumente in der Schema Design-Ansicht bzw. der WSDL Design-Ansicht erstellt, finden Sie im XML-Schema Tutorial und dem WSDL Tutorial.

© 2005 Altova GmbH

5.1 Schema Design-Ansicht

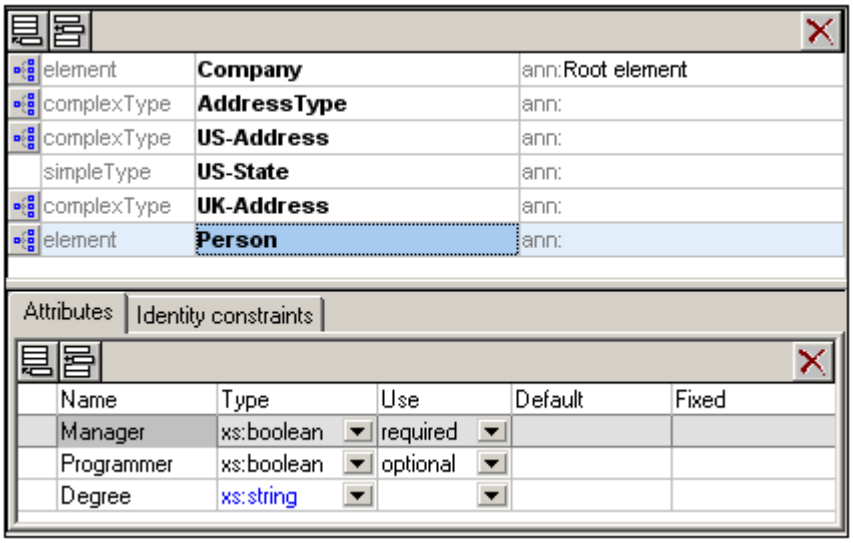
Die Schema Design-Ansicht selbst verfügt über zwei Arten von Ansichten:

- einer Schema-**Hauptübersicht**, in der alle globalen Komponenten (globale Elemente, complexTypes usw.) in einer einfachen Tabelle dargestellt werden.
- Ansichten der Content Models der einzelnen globalen Komponenten (**Content Model-Ansicht**).

Weiter unten erhalten Sie eine kurze Übersicht über die Schema-Übersicht und die Content Model-Ansicht, gefolgt von einer kurzen Beschreibung der in der Schema Design-Ansicht verfügbaren Eingabehilfen. Die beiden Unterkapitel dieses Abschnitts enthalten eine ausführliche Beschreibung der Schema-Übersicht und Content Model-Ansicht.

Hinweis: Mit dem SchemaAgent-Feature von Altova, das Sie über XMLSpy 2005 aufrufen, können Sie Komponenten aus anderen XML-Schemas direkt verwenden. Nähere Informationen dazu finden Sie in dieser Dokumentation im Abschnitt SchemaAgent.

Schema-Übersicht
In der Schema-Übersicht sehen Sie eine Liste aller globalen Komponenten des Schemas (globale Elemente, complexTypes, usw).



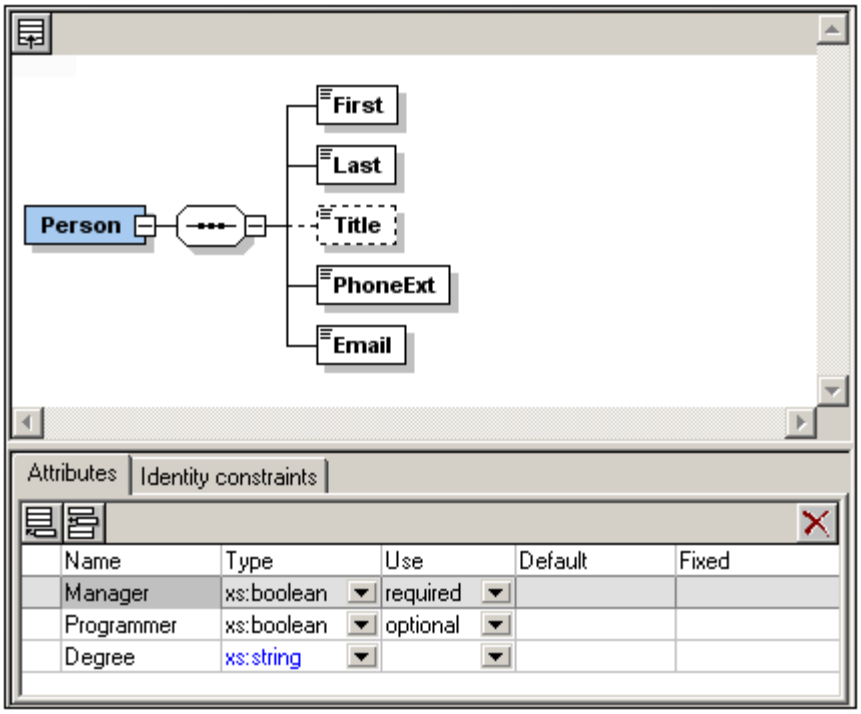
Sie können globale Komponenten einfügen, anhängen oder löschen und ihre Eigenschaften bearbeiten. Verwenden Sie zum Einfügen, Anhängen oder Löschen die entsprechenden Schaltflächen am oberen Rand der Schema-Übersicht. Um Eigenschaften zu bearbeiten, wählen Sie die gewünschte Komponente in der Schema-Übersichtsliste aus und editieren Sie die Eigenschaften entweder über die Eingabehilfen (im rechten Bereich der Ansicht) oder das Attribute-/Identity Constraints- Fenster (im unteren Bereich der Ansicht).

In der Schema-Übersicht stehen Ihnen die folgenden Editierfunktionen zur Verfügung:

- Sie können Komponenten in der Schema-Übersichtsliste mit Drag & Drop verschieben.
- Sie können mit Hilfe der Pfeiltasten navigieren.
- Sie können globale Komponenten, Attribute und Identity Constraints mit Hilfe der Funktionen Ausschneiden/Kopieren-und-Einfügen an eine andere Stelle verschieben und oder von einem Schema in ein anderes verschieben bzw. kopieren.
- Durch Rechtsklick auf eine Komponente öffnen Sie ein Kontextmenü, über das Sie die Annotationsdaten dieser Komponente ausschneiden, kopieren, einfügen oder bearbeiten können.

Content Model-Ansicht
Ein Content Model ist eine Beschreibung der Struktur und des Inhalts eines Elements. Globale Komponenten, die ein Content Model haben können (z.B. Elemente, complexTypes und Model Groups; nicht aber z.b. simpleTypes) werden in der Schema-Übersichtsliste mit einem Symbol links vom Komponentennamen gekennzeichnet. Bei Doppelklick auf dieses Symbol wird das Content Model für diese globale Komponente geöffnet. Alternativ dazu (i) wählen Sie eine Komponente aus und anschließend die Menüoption **Schema-Design | Diagramm anzeigen** oder (ii) doppelklicken Sie im Komponentennavigator (d.h. der Eingabehilfe rechts oben) auf einen Komponentennamen. Beachten Sie, dass immer nur jeweils ein Content Model im Schema geöffnet sein kann. Wenn ein Content Model geöffnet ist, können Sie zum Content Model einer Komponente innerhalb des aktuellen Content Model springen, indem Sie bei gedrückter **Strg**-Taste auf die gewünschte Komponente doppelklicken.

Das Content Model wird in der Content Model-Ansicht als Baumdiagramm (siehe Abbildung unten) dargestellt. Sie können die Darstellungsweise des Diagramms im Dialogfeld "Schema-Ansicht konfigurieren" (Menüeintrag **Schema-Design | Ansicht konfigurieren** einstellen).



Die Content Model-Ansicht verfügt über die folgenden Editierfeatures:

- Jede Ebene (Elemente oder Elementgruppen) im Baumdiagramm ist mittels eines Kompositors mit benachbarten Ebenen verbunden.
- Sie können Objekte des Baumdiagramms (Kompositoren, Elemente, Elementgruppen) mittels Drag-and-Drop verschieben.
- Objekte (Kompositoren, Elemente und Elementgruppen) können über das Kontextmenü (wird durch Rechtsklick auf ein Objekt aufgerufen) hinzugefügt werden.
- Sie können die Eigenschaften eines Objekts in der Details-Eingabehilfe (Kompositoren, Elemente, Elementgruppen) und im Attribute-/Identity Constraints-Fenster bearbeiten.
- Die Attribute und Identity Constraints einer Komponente werden in einem Fenster am unteren Rand des Hauptfensters angezeigt. Attribute und Identity Constraints können auch im Content Model-Diagramm anstatt des Attribute/Identity Constraints-Fensters angezeigt werden. Diese Ansichtsoption lässt sich im Dialogfeld Schema-Ansicht konfigurieren einstellen.

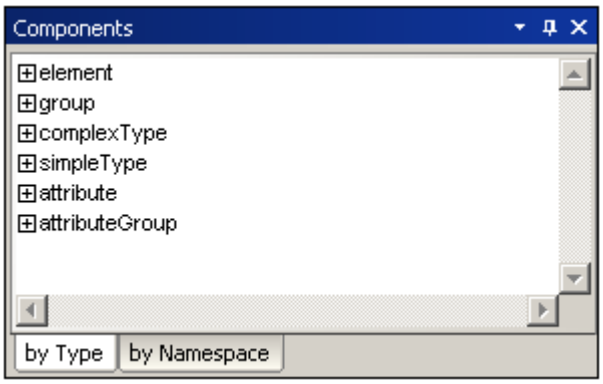
© 2005 Altova GmbH

Diese Funktionen werden in den Unterkapiteln dieses Abschnitts und im Tutorial näher erläutert. Um zur Schema-Übersicht zurückzukehren, klicken Sie auf das Symbol **Globale Elemente anzeigen** oder wählen Sie die Menüoption **Schema-Design | Globale Elemente anzeigen**.

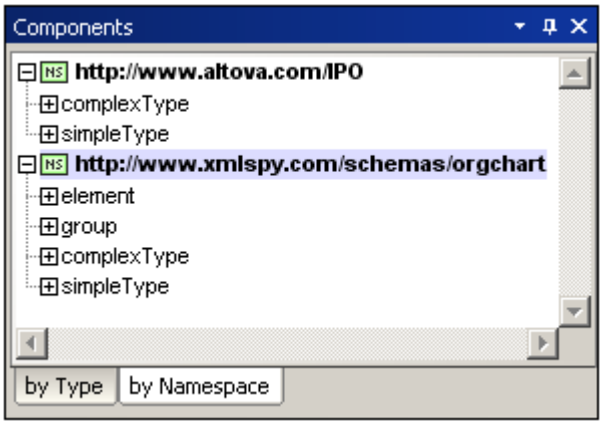
Eingabehilfen in der Schema-Ansicht
Es gibt drei Eingabehilfen in der Schema/WSDL Design-Ansicht: den Komponenten-Navigator, die Details-Eingabehilfe und die Facets-Eingabehilfe. Im Folgenden sind diese näher beschrieben.

Komponenten-Navigator
Der Komponenten-Navigator ist eine Eingabehilfe in der **Schema/WSDL Design-Ansicht**. Er hat zwei Aufgaben:

- Globale Komponenten in einem Baumdiagramm nach Komponententyp und Namespace (*siehe Abbildung unten)* zu strukturieren. Dadurch steht Ihnen eine strukturierte Ansicht aller globalen Komponenten zur Verfügung.
- Ihnen das Navigieren zur Content Model-Ansicht einer globalen Komponenten zu ermöglichen und diese anzuzeigen - falls die Komponente ein Content Model hat. Wenn eine Komponente kein Content Model hat, wird die Komponente in der Schema-Ansicht markiert. Auch aus anderen Schemas inkludierte oder importierte globale Komponenten werden im Komponenten-Navigator angezeigt.



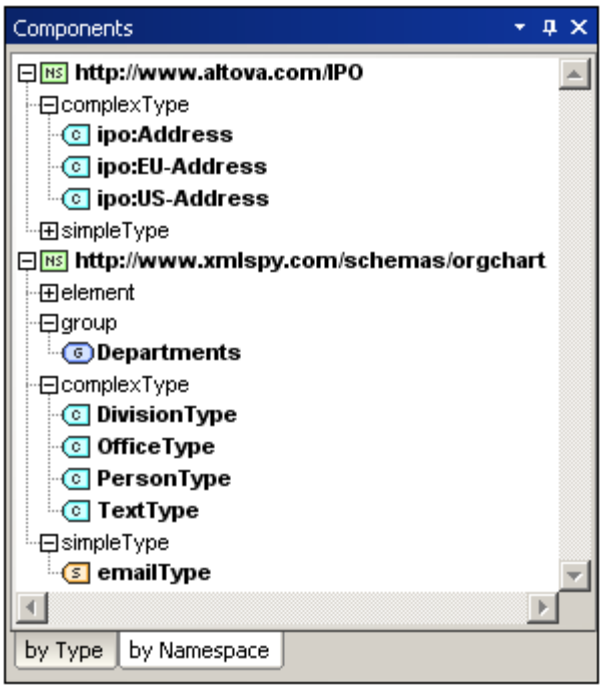
Auf dem Register "by Type" (*siehe oben)* werden die globalen Komponenten nach ihrem Komponententyp in einer Baumstruktur gegliedert. Auf dem Register Namespace (*siehe unten)* sind die Komponenten zuerst nach Namespace und anschließend nach Komponententyp geordnet. Beachten Sie, dass ein Komponententyp nur dann in einer Baumstruktur angezeigt wird, wenn mindestens eine Komponente dieses Typs im Schema vorhanden ist.



In der Baumstruktur werden globale Komponenten in die folgenden sechs Gruppen eingeordnet:

- Element-Deklarationen (Elemente)
- Model Groups (Groups)
- ComplexTypes
- SimpleTypes
- Attribut-Deklarationen (Attribute)
- Attribut-Gruppen

Wenn Sie eine Komponenten-Typ-Gruppe in der Baumstruktur erweitern, werden alle Komponenten in dieser Gruppe angezeigt (*siehe Abbildung unten)*. Auf diese Art können Sie schnell zur gewünschten Komponente navigieren.



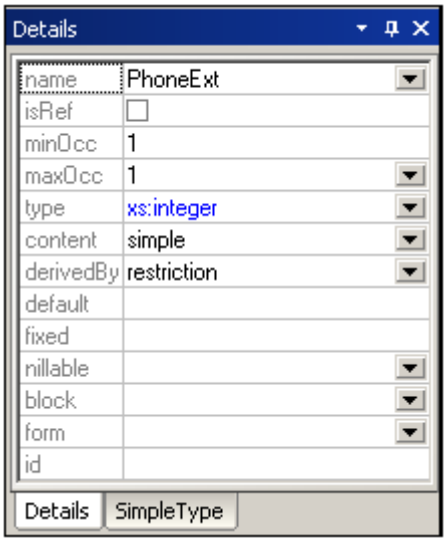
Wenn eine Komponente ein Content Model hat (d.h. wenn es sich um ein Element, eine Group oder einen complexType handelt), wird mit einem Doppelklick darauf das Content Model dieser Komponente in der Content Model-Ansicht (im Hauptfenster) angezeigt. Wenn die Komponente kein Content Model hat (d.h. wenn es sich um einen simpleType, ein Attribut oder eine Attribut-Gruppe handelt), so wird die Komponente in der Schema-Übersicht (im Hauptfenster) markiert.

 © 2005 Altova GmbH

Hinweis: Wenn es sich bei der Komponente um ein inkludiertes oder importiertes Schema handelt, wird dieses geöffnet (wenn es nicht bereits offen ist) und es wird entweder das Content Model der Komponente in der Content Model-Ansicht angezeigt oder die Komponente wird in der Schema-Übersicht markiert.

Details-Eingabehilfe

Die Details-Eingabehilfe steht in der **Schema/WSDL Design-Ansicht** zur Verfügung. Sie enthält editierbare Informationen über den im Hauptfenster ausgewählten Kompositor/die ausgewählte Komponente. Wenn Sie eine Schemadatei, die Datenbank-Extensions enthält, bearbeiten, wird unter Umständen ein zusätzliches Register mit Informationen über die DB Extensions angezeigt. Derzeit werden Oracle-, SQL Server- und Tamino-Datenbanken unterstützt.

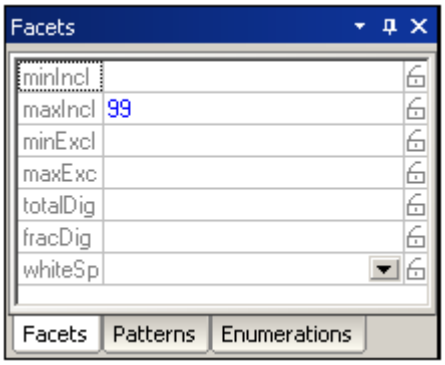


Um die Eigenschaften des derzeit ausgewählten Kompositors oder der ausgewählten Komponente zu ändern, doppelklicken Sie auf das zu editierende Feld und bearbeiten Sie es oder geben Sie Text direkt ein. Wenn in dem gewünschten Feld eine Dropdown-Liste zur Verfügung steht, wählen sie den gewünschten Wert aus; dieser Wert wird in dem Feld übernommen.

Änderungen, die Sie über die Details-Eingabehilfe vornehmen, werden sofort im Content Model-Diagramm angezeigt.

Facets-Eingabehilfe

Die Facets-Eingabehilfe steht in der **Schema/WSDL Design-Ansicht** zur Verfügung und gestattet die Eingabe der Werte von Facets, Patterns und Enumerations. Ein Beispiel zur Verwendung von Facets und Enumerations in einem XML-Schema finden Sie in den entsprechenden Abschnitten des Tutorials.



Um Facets, Patterns oder Enumerations in der Facets-Eingabehilfe zu ändern, gehen Sie vor wie folgt:

1. Wählen Sie das entsprechende Register aus (Facets, Patterns oder Enumerations)
2. Wählen Sie einen Wert aus der Dropdown-Liste aus, falls vorhanden. Doppelklicken Sie andernfalls auf eine Zeile und bearbeiten Sie den Text manuell bzw. geben Sie ihn manuell ein.

Hinweis: Sie können die Patterns und Enumerations einer Komponente mit Hilfe der Tastenkürzel zum Ausschneiden, Kopieren und Einfügen (Strg+X, Strg+C bzw. Strg+V) in eine andere Komponente kopieren. Wählen sie in der Facets-Eingabehilfe ein oder mehrere Patterns oder Enumerations aus, um diese auszuschneiden oder zu kopieren, klicken Sie anschließend in die Facets-Eingabehilfe der Zielkomponente und fügen Sie sie ein.

5.1.1 Schema-Übersicht

Auf der obersten Ebene eines XML-Dokuments (d.h. auf Ebene der Child-Elemente des `schema`-Elements) können die folgenden fünf Basiskomponenten definiert werden:

- Annotation
- Typ-Definition (simple oder complex)
- Deklaration (Element oder Attribut)
- Attribute Group
- Model Group

Diese Komponenten auf der obersten Ebene werden **globale Komponenten** genannt. In der **Schema-Übersicht** werden allen globen Komponenten in Ihrem Schema in Tabellenform aufgelistet. Einige globale Komponenten (wie z.B. complexTypes, Element-Deklarationen und Model Groups) können ein Content Model haben, das Struktur und Inhalt der Komponente beschreibt. Andere globale Komponenten (wie z.B. Annotationen, simpleTypes und Attribut-Gruppen) haben kein Content Model. Komponenten, die ein Content Model haben können, sind durch ein Symbol links vom Komponentennamen gekennzeichnet. Wenn Sie auf dieses Symbol klicken, wird die Content Model-Ansicht für diese globale Komponente angezeigt.

Grundlegende Begriffe

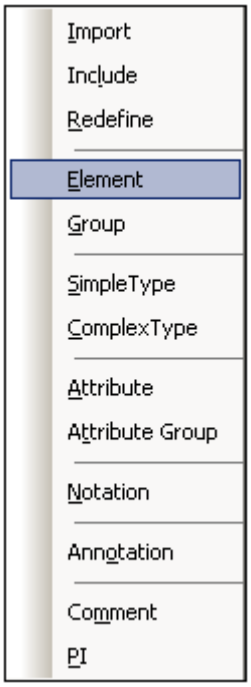
- *SimpleType und complexType.* Ein simpleType dient dazu, alle Attribute und Elemente zu definieren, die nur Text und kein Attribut enthalten. Ein simpleType hat daher kein Content Model, sondern nur Text (der durch den Datentyp eingeschränkt werden kann). Ein complexType hat dagegen mindestens ein Child-Element oder -Attribut. Wenn Sie ein Child-Element für ein Element deklarieren, wird diesem automatisch der Typ complexType zugewiesen.
- *Globale und lokale Komponenten.* Bei einer globalen Komponente kann es sich um jede

 © 2005 Altova GmbH

der fünf oben angeführten Arten handeln. Eine globale Komponente kann in der Schema-Übersicht definiert werden und scheint daraufhin sofort in der Liste der globalen Komponenten in der Schema-Übersicht auf. Wenn es sich bei der globalen Komponente um einen complexType, eine Element-Deklaration oder eine Model Group handelt, können Sie anschließend ihr Content Model definieren, indem Sie es in der Content Model-Ansicht bearbeiten. Sobald eine globale Komponente definiert ist, kann sie von lokalen Komponenten referenziert werden. Eine lokale Komponente wird direkt im Content Model einer Komponente erstellt. Beachten Sie, dass eine lokale Komponente in der Content Model-Ansicht (über das Kontextmenü) in eine globale Komponente konvertiert werden kann.

Erstellen globaler Komponenten
So erstellen Sie in der Schema-Übersicht eine globale Komponente:

1. Klicken Sie oben in der Schema-Übersicht auf das Symbol "Einfügen" oder "Anhängen" . Daraufhin wird ein Menü angezeigt, aus dem Sie verschiedene Komponententypen (Element, simple type, complex type, model group usw.) auswählen können.



2. Wählen Sie den gewünschten Komponententyp aus. In der Liste der globalen Komponenten wird ein Eintrag dieses Typs erstellt.
3. Geben Sie den Namen der Komponente hier ein und drücken Sie die **Eingabetaste**. Der Name der neuen globalen Komponente wird zur entsprechenden Liste bzw. zu den entsprechenden Listen (Elm, Grp, Com, Sim, usw.) in der Komponenten-Navigator-Eingabehilfe hinzugefügt. Sie können das Content Model der neuen globalen Komponente entweder durch Doppelklicken auf den Komponentennamen im Komponenten-Navigator oder durch Klicken auf das Symbol links vom Komponentennamen in der Liste der globalen Komponenten bearbeiten.

Hinweis:

- Sie können außerdem eine globale Komponente erstellen, während Sie in der Content Model-Ansicht arbeiten. Klicken Sie mit der rechten Maustaste auf eine beliebige Stelle

im Fenster und wählen Sie **Neues Global | Element**.

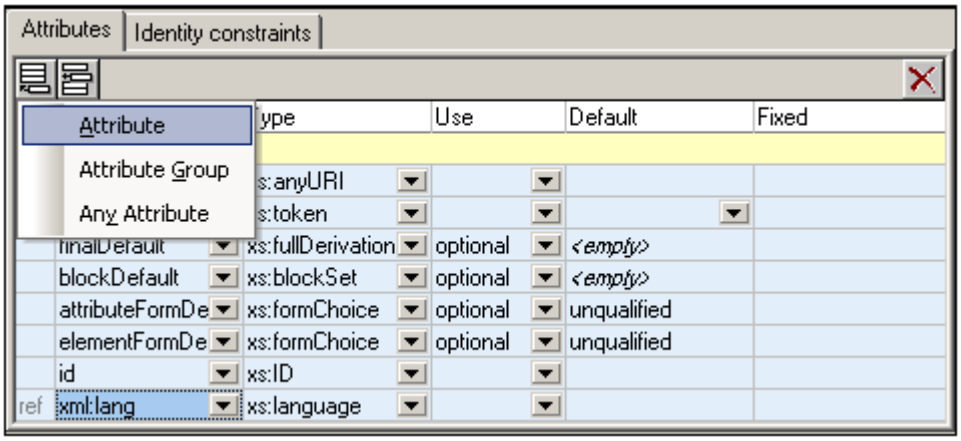
- Beim Bearbeiten in der Content Model-Ansicht können Sie ein lokales Element zu einem globalen - oder sogar zu einem complexType machen, wenn das Element ein Child-Element oder -Attribut hat. Wählen Sie das lokale Element aus, rechtsklicken Sie in das Fenster und wählen Sie **Als global definieren | Element** oder **Als global definieren | Complex Type**.

Löschen globaler Komponenten
So löschen Sie eine globale Komponente:

1. Wählen Sie die globale Komponente in der Schema-Übersicht in der Liste der globalen Komponenten aus.
2. Drücken Sie die **Entfernen**-Taste oder klicken Sie am oberen Rand der Schema-Übersicht auf das Symbol **Löschen** .

Attribute und Identity Constraints von Komponenten
Sie können Attribute und Identity Constraints für Komponenten entweder in der Schema-Übersicht oder in der Content Model-Ansicht definieren. In der Schema-Übersicht werden die Attribute und Identity Constraints einer Komponente im Attribute-/Identity Constraints-Fenster im unteren Bereich des Schema-Übersichtsfensters angezeigt und können hier editiert werden. In der Content Model-Ansicht können Attribute und Identity Constraints entweder im Attribute-/Identity Constraints-Fenster am unteren Rand des Content Model-Fensters oder innerhalb des Content Model-Diagramms selbst angezeigt werden, wo Sie diese bearbeiten können. Über eine Einstellung im Dialogfeld Schema-Ansicht konfigurieren, die Sie über den Menübefehl **Schema-Design | Ansicht konfigurieren** aufrufen, können Sie einstellen, wie Attribute und Identity Constraints in der Content Model-Ansicht angezeigt werden sollen.

Definieren von Attributen für eine Komponente
Attribute für eine Komponente lassen sich über das Attribute-/Identity Constraints-Fenster am unteren Rand des Schema-Übersichtsfensters definieren.



So definieren Sie Attribute für eine globale Komponente, für die Attribute zulässig sind:

1. Wählen Sie die globale Komponente in der Liste der globalen Komponenten aus.
2. Wählen Sie im Attribute-/Identity Constraints-Fenster das Register "Attribute".
3. Klicken Sie auf das Symbol "Anhängen" oder "Einfügen" links oben auf dem Register "Attribute".
4. Wählen Sie im Popup-Menü, das daraufhin angezeigt wird, den gewünschten Attributtyp

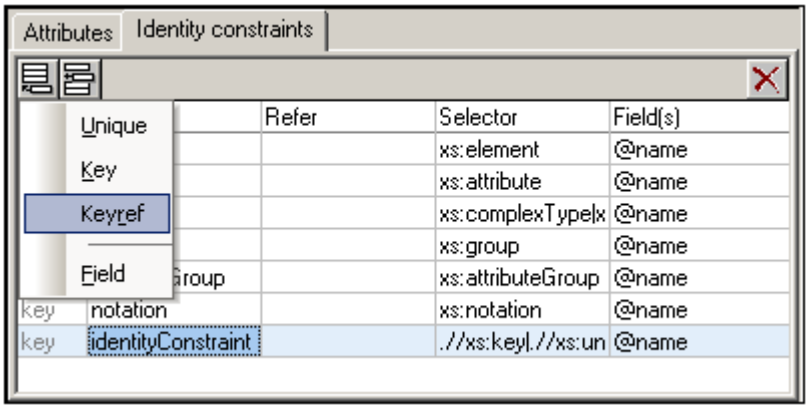
 © 2005 Altova GmbH

aus. In der Attributliste wird ein Eintrag erstellt.
5. Geben Sie in dem soeben erstellten Eintrag die Eigenschaften des Attributs an.

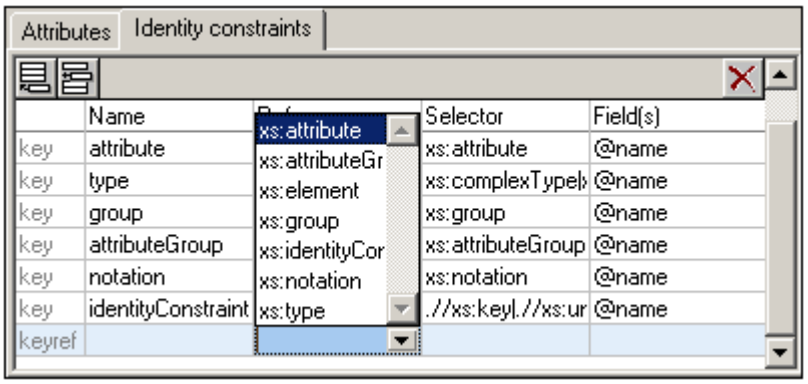
Hinweis: Sie können Attribute für globale Komponenten auch in der Content Model-Ansicht definieren. Wählen sie die globale Komponente aus und definieren Sie anschließend Attribute wie oben beschrieben.

Definieren von Identity Constraints für eine Komponente
Zum Definieren von Identity Constraints für eine Komponente wird das Attribute-/Identity Constraints-Fenster im unteren Bereich des Schema-Übersichtsfensters verwendet. So definieren Sie Identity Constraints für eine Komponente:

1. Wählen Sie die globale Komponente in der Liste der globalen Komponenten aus.
2. Wählen Sie im Attribute-/Identity Constraints-Fenster das Register "Identity Constraints" aus.
3. Klicken Sie auf das Symbol "Anhängen" oder "Einfügen" in der linken oberen Ecke des Registers.
4. Wählen Sie im Popup-Fenster die Art des gewünschten Identity Constraint aus.



In der Liste der Identity Constraints wird ein Eintrag erstellt.
5. Geben Sie in diesen Eintrag die Eigenschaften des Identity Constraint ein. `Selector` gibt den Nodeset an, für den der Identity Constraint gilt. `Field` identifiziert den Wert, der innerhalb des von der `Selector`-Eigenschaft identifizierten Nodesets eindeutig sein muss. Wenn Sie als Identity Constraint Typ `Keyref` ausgewählt haben, ist die `Refer`-Eigenschaft aktiviert und in der Dropdown-Liste werden alle im Schema definierten Schlüssel aufgelistet (*siehe Abbildung unten*).



Hinweis: Sie können auch für globale Komponenten in der Content Model-Ansicht Identity

Constraints definieren. Wählen Sie die globale Komponente aus und definieren Sie wie oben beschrieben Identity Constraints.

5.1.2 Content Model-Ansicht

In der **Content Model-Ansicht** können Sie das Content Model der folgenden drei Komponententypen grafisch mit wenigen Mausklicks definieren:

- ComplexTypes,
- Element-Deklarationen und
- Model Goups

Alle anderen Schema-Komponenten (Annotationen, Attribut-Deklarationen, simpleTypes usw.) haben kein Content Model.

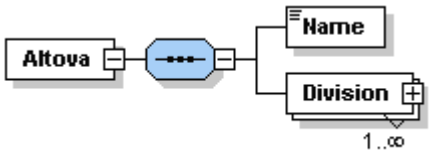
In der Content Model-Ansicht werden die verschiedenen Teile des Content Model grafisch dargestellt. Diese Teile lassen sich in zwei große Gruppen unterteilen: **Kompositoren** und **Komponenten**. Normalerweise wird ein Kompositor hinzugefügt und anschließend die gewünschten Child-Komponenten.

Kompositoren
Ein **Kompositor** definiert die Reihenfolge, in der Child-Elemente angeordnet sind. Es gibt drei Kompositoren: `sequence`, `choice` und `all`. So fügen Sie einen Kompositor ein:

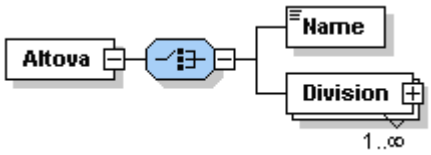
1. Rechtsklicken Sie auf das Element, zu dem Sie Child-Elemente hinzufügen möchten.
2. Wählen Sie den Befehl **Child hinzufügen | Sequence** (oder **Choice** oder **All**).

Daraufhin wird ein Kompositor hinzugefügt, der folgendermaßen aussieht:

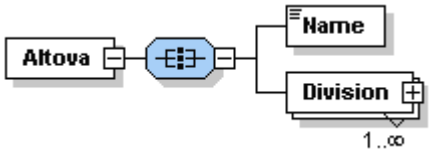
- **Sequence**



- **Choice**



- **All**



Um den Kompositor zu ändern, rechtsklicken Sie darauf und wählen Sie **Modell ändern | Sequence** (oder **Choice** oder **All**). Nachdem Sie den Kompositor hinzugefügt haben, müssen Sie (ein) Child-Element(e) oder eine Model Group hinzufügen.

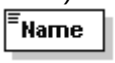
© 2005 Altova GmbH

Komponenten im Content Model
Unten finden Sie eine Liste von Komponenten, die in Content Models verwendet werden. Die grafische Darstellung der einzelnen Komponenten gibt Ihnen genaue Informationen über die Art der Komponente und ihre strukturellen Eigenschaften.

- Obligatorisches Einzelelement

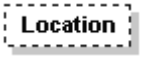
  Details: Das Rechteck zeigt an, dass es sich um ein Element handelt, die durchgezogene Umrandung bedeutet, dass das Element "required" also obligatorisch ist. Das Fehlen eines Zahlenbereichs weist darauf hin, dass es ein Einzelelement ist (d.h. `minOcc=1` und `maxOcc=1`). Der Name des Elements ist `Country`. Die blaue Farbe zeigt an, dass das Element gerade ausgewählt ist (eine Komponente wird durch Anklicken ausgewählt). Wenn eine Komponente nicht ausgewählt ist, ist sie weiß.

- Obligatorisches Einzelelement, das geparste Zeichendaten enthält (d.h. ein Child Text-Node)

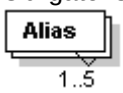
  Details: Rechteck = Element; Durchgezogene Umrandung = obligatorisch; fehlender Zahlenbereich = `minOcc=1` und `maxOcc=1`; Textzeilensymbol in der linken oberen Ecke = ein Child Text-Node (Der Content-Typ könnte "simple content" (nur Text-Node) oder "mixed content" (Text und Elemente) sein; in dieser Abbildung handelt es sich um "simple content", da kein Plus-Zeichen vorhanden ist, das anzeigt, dass das Element weiter erweitert werden kann, um ein Content Model anzuzeigen. Der Element-Name ist `Name`.

- Optionales Einzelelement

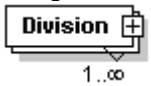
  Details: Rechteck = Element; gestrichelte Umrandung = optional; fehlender Zahlenbereich = `minOcc=0` und `maxOcc=1`. Der Element-Name ist `Location`.
  Hinweis: Im Kontextmenü können Sie ein obligatorisches Element über die Menüoption **Optional** in ein optionales umwandeln.

- Obligatorisches mehrmals vorkommendes Element

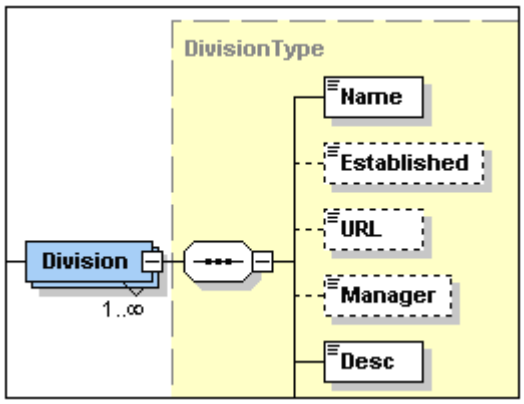
  Details: Rechteck = Element; durchgezogene Umrandung = obligatorisch; Zahlenbereich 1..5 = `minOcc=1` und `maxOcc=5`. Der Element-Name ist `Alias`.

- Obligatorisches mehrfach vorkommendes Element, das Child-Elemente enthält

  Details: Rechteck = Element; durchgezogene Umrandung = obligatorisch; Zahlenbereich 1..unendlich = `minOcc=1` und `maxOcc=unbounded`; Plus-Zeichen = complex content (d.h. es hat mindestens ein Child-Element oder -Attribut). Der Element-Name ist `Division`.
  Hinweis: Über die Kontext-Menüoption **Unbounded** können Sie `maxOcc` in `unbounded`

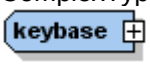
ändern.
Wenn Sie auf das +-Zeichen des Elements klicken, wird die Baumstruktur erweitert und Sie sehen die Child-Elemente.



- Element, das ein globales Element referenziert

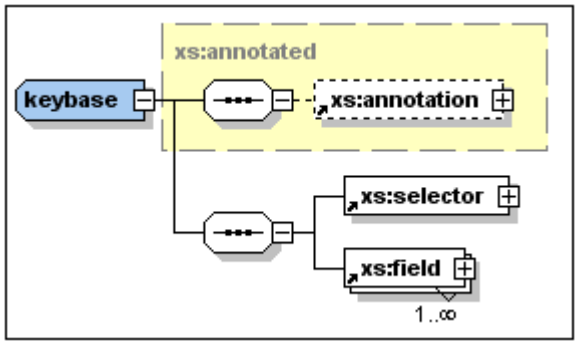
Details: Pfeil in der linken unteren Ecke = Element, das ein globales Element referenziert; Rechteck = Element; durchgezogene Umrandung = obligatorisch; Zahlenbereich 1..unendlich = `minOcc=1` und `maxOcc=unbounded`; Plus-Zeichen = complex content (d.h. mindestens ein Child-Element oder -Attribut). Der Element-Name ist `xs:field`.
Hinweis: Ein globales Element kann von simple- und complex type-Definitionen aus referenziert werden, sodass Sie eine globale Deklaration an mehreren Stellen in Ihrem Schema wiederverwenden können. Es gibt zwei Möglichkeiten, um eine Referenz auf ein globales Element zu erstellen: (i) durch Eingabe eines Namens für das lokale Element, der mit dem des globalen Elements identisch ist; und (ii) durch Rechtsklick auf das lokale Element und Auswahl der Kontextmenüoption **Reference**. Um die Definition eines globalen Elements anzusehen, halten Sie die **Strg**-Taste gedrückt und doppelklicken Sie auf das Element. Alternativ dazu können Sie auch rechtsklicken und den Befehl **Gehe zu Definition** auswählen. Wenn Sie ein Element referenzieren, das nicht existiert, wird der Elementname rot angezeigt. Dies ist eine Warnung, dass es keine referenzierbare Definition gibt.

- ComplexType



Details: Ein unregelmäßiges Sechseck mit einem Plus-Zeichen (siehe oben) kennzeichnet einen complexType. Der hier dargestellte complexType hat den Namen `keybase`. Dieses Symbol bedeutet, dass es sich um einen globalen complexType handelt. Ein globaler complexType wird in der Schema-Übersicht deklariert. Sein Content Model wird normalerweise in der Content Model-Ansicht definiert. Ein globaler complexType kann für folgende Zwecke verwendet werden: (i) als Datentyp eines Elements oder (ii) als Base Type eines anderen complexType, indem Sie ihn in der Details-Eingabehilfe (entweder in der Content Model-Ansicht oder in der Schema-Übersicht) dem Element bzw. complexType zuweisen.

© 2005 Altova GmbH



Der oben dargestellte complexType `keybase` wurde in der Schema-Übersicht mit einem Base Type `xs:annotated` deklariert. Der Base Type wird als Rechteck mit einer grauen gestrichelten Umrandung und gelbem Hintergrund dargestellt. Anschließend wurden in der Content Model-Ansicht die Child-Elemente `xs:selector` und `xs:field` erstellt. (Beachten Sie die kleinen Pfeile in der linken unteren Ecke der Rechtecke mit der Beschriftung `xs:selector` und `xs:field`. Diese Pfeile bedeuten, dass beide Elemente globale Elemente dieses Namens referenzieren.)

Ein lokaler complexType wird direkt in der Content Model-Ansicht durch Erstellung eines Child-Elements oder -Attributs für ein Element definiert. Es gibt kein separates Symbol für lokale complexTypes.

Hinweis: Der Base Type eines Content Models wird als Rechteck mit grauer gestrichelter Umrandung und gelber Hintergrundfarbe dargestellt. Um das Content Model des Base Type aufzurufen, doppelklicken Sie auf seinen Namen.

- Model Group

Details: Das unregelmäßige Achteck mit einem Plus-Zeichen (siehe Abbildung oben) kennzeichnet eine Model Group. Eine Model Group gestattet Ihnen, Element-Deklarationen zu definieren und wiederzuverwenden.
Hinweis: Wenn die Model Group (in der Schema-Übersicht) deklariert wird, erhält sie einen Namen. Anschließend wird (in der Content Model-Ansicht) der Content Mode durch Zuweisung eines Child-Kompositors, der die Element-Deklarationen enthält, definiert. Wenn die Model Group verwendet wird, wird Sie (in der Content Model-Ansicht) innerhalb des Content Models einer anderen Komponente als Child eingefügt oder angehängt.

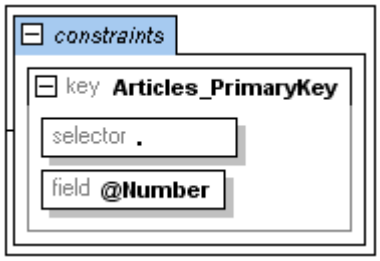
- Wildcards

Details: Das unregelmäßige Achteck mit `any` auf der linken Seite kennzeichnet eine Wildcard.
Hinweis: Wildcards werden als Platzhalter für Elemente verwendet, die im Schema nicht definiert sind oder aus anderen Namespaces stammen. `##other` = Elemente können zu jedem Namespace gehören, bei dem es sich nicht um den im Schema definierten Target Namespace handelt; `##any` = Elemente können zu jedem Namespace gehören; `##targetNamespace` = Elemente müssen zum Target Namespace gehören, der im Schema definiert wurde; `##local` = Elemente können nicht zu jedem beliebigen Namespace gehören; *`anyURI`* = Elemente gehören zu dem von Ihnen angegebenen Namespace.

- Attribute

Details: Attribute werden durch das kursiv geschriebene Wort *'attributes'* in einem erweiterbaren Rechteck dargestellt. Jedes Attribut wird in einem Rechteck mit einer strichlierten Umrandung dargestellt.
Hinweis: Attribute können in der Details-Eingabehilfe bearbeitet werden. Sie können Attribute im Diagramm der Content Model-Ansicht oder in einem Fenster unterhalb der Content Model-Ansicht anzeigen. Durch Klicken auf das Symbol "Attribute direkt in Diagramm anzeigen" können Sie zwischen den beiden Ansichten hin- und herwechseln. Wenn Attribute im Diagramm der Content Model-Ansicht angezeigt werden, werden alle Attribute eines Einzelelements in einem Rechteck mit gestrichelter Umrandung dargestellt. Um die Reihenfolge von Attributen eines Elements zu ändern, ziehen Sie das Attribut aus dem Kästchen heraus und lassen Sie die Maustaste los, wenn an der gewünschten Stelle ein Pfeil erscheint.

- Identity Constraints



Details: Identity Constraints werden durch das kursiv gedruckte Wort '*constraints*' in einem erweiterbaren Rechteck dargestellt.
Hinweis: Die im Content Model einer Komponente aufgelisteten Identity Constraints zeigen Constraints, die mit Hilfe der Elemente `key` und `keyref` definiert wurden und das Element `unique` haben. Identity Constraints, die mit Hilfe des `ID`-Datentyps definiert wurden, werden im Content Model-Diagramm nicht angezeigt, jedoch in der Details-Eingabehilfe. Identity Constraints können in der Content Model-Ansicht oder in einem Fenster unterhalb der Content Model-Ansicht angezeigt werden. Durch Klicken auf das Symbol "Contraints direkt im Diagramm anzeigen" können Sie zwischen den beiden Ansichten hin- und herwechseln.

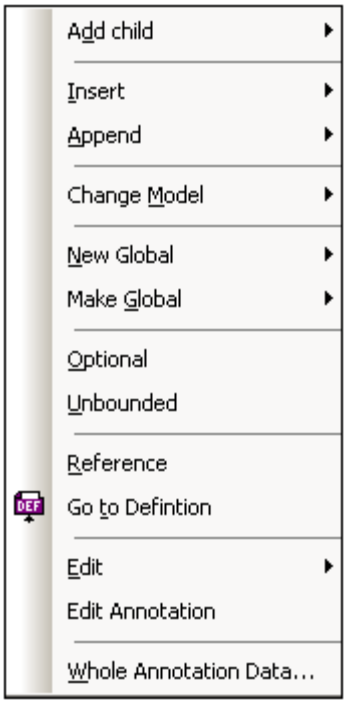
Hinweis:

- Durch Klicken auf das Symbol "Vordefinierte Details hinzufügen" können Sie Eigenschaften-Beschreibungen, die Sie im Dialogfeld Schema-Ansicht konfigurieren definiert haben, ein oder ausblenden.
- Durch Auswahl der Symbole "Attribute direkt im Diagramm anzeigen" und "Constraints direkt im Diagramm anzeigen" können Sie Attribute und Identity Constraints wahlweise entweder im Content Model-Diagramm selbst oder in einem Fenster unterhalb der Content Model-Ansicht anzeigen lassen.
- In der Content Model-Ansicht können Sie zur Content Model-Ansicht einer beliebigen globalen Komponente im aktuellen Content Model springen, indem Sie die **Strg**-Taste

© 2005 Altova GmbH

gedrückt halten und auf die gewünschte Komponente doppelklicken. Durch Doppelklicken auf den Namen eines Base Type können Sie das Content Model dieses Base Type aufrufen.

Andere Editiermöglichkeiten in der Content Model-Ansicht
Die Editierbefehle für die Content Model-Ansicht können über das Kontextmenü (*siehe Screenshot unten)* aufgerufen werden, das Sie durch einen Rechtsklick in die Content Model-Ansicht öffnen können. Im Folgenden wird beschrieben, welche Funktionen Ihnen zur Verfügung stehen.



Hinzufügen von Child-Kompositoren/Komponenten und Einfügen/Anhängen von Kompositoren/Komponenten

1. Rechtsklicken Sie auf den Kompositor bzw. die Komponente. Daraufhin wird das Kontextmenü (in dem nur die zulässigen Befehle aktiv sind) geöffnet.
2. Wählen Sie den gewünschten Befehl aus.

Ändern eines Kompositors

1. Rechtsklicken Sie auf den gewünschten Kompositor.
2. Wählen Sie die Kontextmenüoption **Modell ändern** und wählen Sie im Untermenü den Kompositor aus, den Sie verwenden möchten. (Der aktuell ausgewählte Kompositor ist durch ein Häkchen markiert.) Wenn ein Kompositor an dieser Stelle nicht verwendet werden darf, ist er ausgegraut.

Erstellen globaler Komponenten

- Um eine globale Komponente zu erstellen, rechtsklicken Sie an eine beliebige Stelle in der Content Model-Ansicht, wählen Sie den Befehl **Neues Global** und aus dem Untermenü die gewünschte Komponente aus.

- Um ein lokales Element zu einem globalen Element oder complexType zu machen, rechtsklicken Sie auf das lokale Element, wählen Sie **Als global definieren** und wählen Sie im Untermenü entweder **Element** oder **Complex type** aus. Wenn eine dieser Komponenten an dieser Stelle nicht erstellt werden kann, ist sie ausgegraut.

Ändern der "Occurrence"-Definition
Sie können zwischen einem "minimum occurrence" und "maximum occurrence" Wert eines Kompositors zwischen `0` und `1` (für `minOccurs`) bzw. `1` und `unbounded` (für `maxOccurs`) wechseln. Gehen Sie dazu vor wie folgt:

1. Rechtsklicken Sie auf den Kompositor oder die Komponente, für die der Occurence-Wert geändert werden muss.
2. Wählen Sie die Kontextmenüoption **Optional**, um den `minOccurs`-Wert zwischen `0` und `1` zu wählen. Wählen Sie die Menüoption **Unbounded,** um den `maxOccurs` Wert zwischen `1` und `unbounded` einzustellen. Die jeweils ausgewählte Option (mit dem Wert `0` bzw. `unbounded`) ist links neben dem Menüeintrag durch ein Häkchen gekennzeichnet.

Wechseln zwischen einer lokalen und einer globalen Definition
Wenn ein globales Element vorhanden ist, das denselben Namen wie ein lokales Element hat, können Sie die globale Definition referenzieren oder die lokale Definition verwenden. Gehen Sie dabei vor wie folgt:

1. Rechtsklicken Sie auf das Element.
2. Wählen Sie die Kontextmenüoption **Reference**. Wenn das globale Element referenziert wird, wird dieser Menüeintrag mit einem Häkchen versehen. Wenn die lokale Definition verwendet wird, erhält der Menüeintrag **Reference** kein Häkchen.

Zu einer Definition springen
Sie können von einem Content Model zur Definition jeder globalen Komponente springen, die in diesem Content Model enthalten ist. Gehen Sie dazu vor wie folgt:

1. Rechtsklicken Sie auf die globale Komponente. Dabei kann es sich um das gelbe Rechteck eines Base Type, ein Element, das ein globales Element referenziert oder eine Model Group handeln.
2. Wählen Sie die Kontextmenüoption **Gehe zu Definition**. Daraufhin wird die Content Model-Ansicht der globalen Komponente geöffnet.

Alternativ dazu doppelklicken Sie auf den Namen des Base Type oder drücken Sie die Strg-Taste und doppelklicken Sie auf das referenzierende Element oder die Model Group.

Bearbeiten von Elementnamen

1. Rechtsklicken Sie auf das Element.
2. Wählen Sie die Kontextmenüoption **Bearbeiten | Name** und ändern Sie den Namen.

Alternativ dazu können Sie auch auf den Elementnamen doppelklicken und die Änderung eingeben.

Erstellen und Bearbeiten von Dokumentation für einen Kompositor oder eine Komponente
Sie können zu einzelnen Kompositoren und Komponenten als Hilfe für Schema-Editoren Dokumentation hinzufügen. Gehen Sie dazu vor wie folgt:

© 2005 Altova GmbH

1. Rechtsklicken Sie auf den Kompositor bzw. die Komponente.

2. Wählen Sie die Kontextmenüoption **Annotation in Diagramm bearbeiten**. Daraufhin wird der Raum unterhalb des Kompositors/der Komponente markiert und Sie können beschreibenden Text zum Kompositor bzw. der Komponente eingeben. In der Textansicht wurden dadurch die Elemente `annotation` und `annotation/documentation` erstellt. Das Element `documentation` enthält den von Ihnen eingegebenen beschreibenden Text.

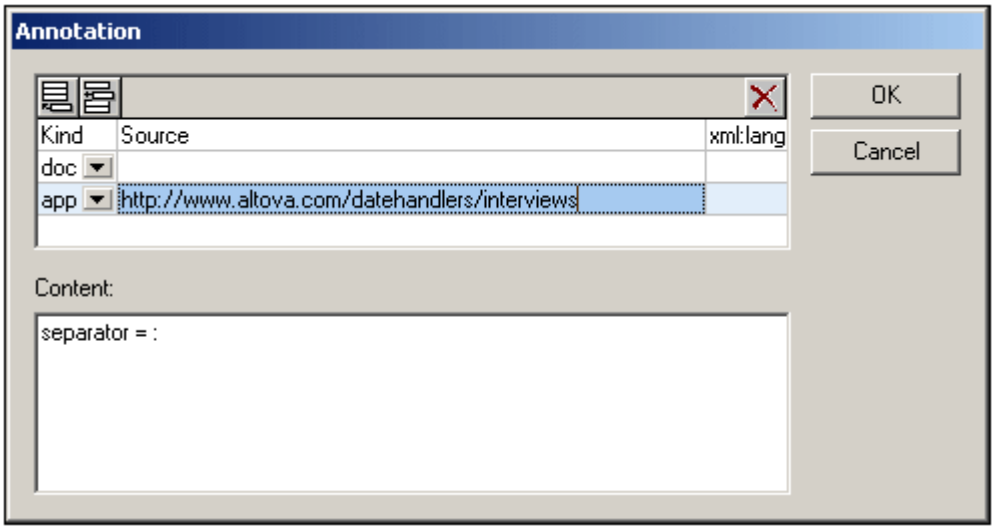
Alternativ dazu können Sie auf den Kompositor/die Komponente rechtsklicken und die Menüoption "Alle Annotationsdaten" wählen. Im Dialogfeld "Annotation", das daraufhin angezeigt wird, können Sie ein Dokumentationselement anhängen oder einfügen und Inhalt dafür eingeben.

Um bereits vorhandenen Dokumentationstext zu bearbeiten, können Sie eine der oben beschriebenen Methoden verwenden, schneller ist es jedoch, wenn Sie im Diagramm auf die Annotation doppelklicken und diese direkt bearbeiten.

Erstellen und Bearbeiten von Applikationsinformationen für einen Kompositor oder eine Komponente

1. Rechtsklicken Sie auf den Kompositor oder die Komponente.

2. Wählen Sie die Kontextmenüoption **Alle Annotationsdaten**. Damit wird das Dialogfeld "Annotation" (*siehe Abbildung unten)* aufgerufen. Wenn für dieses Element eine Annotation (entweder documentation oder appinfo) vorhanden ist, sehen Sie dafür im Dialogfeld eine entsprechende Zeile.



3. Um ein `appinfo`-Element zu erstellen, klicken Sie auf das Symbol "Anhängen" oder "Einfügen" in der linken oberen Ecke, um eine neue Zeile anzuhängen bzw. einzufügen.
4. Wählen Sie im Feld "Kind" der neuen Zeile die Option `app` aus der Dropdown-Liste aus.

5. Geben Sie im Content-Bereich des Dialogfelds das Skript oder die Informationen ein, die von der Applikation verarbeitet werden sollen.
6. Optional dazu können Sie im Feld "Source" eine Quell-URI eingeben, unter der die verarbeitende Applikationen weitere Informationen abrufen kann.

Informationen zu XML-Schema-Annotationen
XML-Schema-Annotationen werden im `annotation`-Element gespeichert. Es gibt zwei Arten von Annotationen. Beide sind Elemente des annotation-Elements:

- Kompositor- oder Komponenten-Dokumentation: Enthält Informationen, die für Schema-Editoren nützlich sein könnten. Diese Informationen befinden sich im `documentation` Child-Element von `annotation`.
- Applikationsinformationen: Hier können ein Skript oder Daten eingefügt werden, die von einer verarbeitenden Applikation gelesen werden können; diese Informationen befinden sich im `appinfo`-Child-Element von `annotation`..

Unterhalb finden Sie den Text eines annotation-Elements. Er beruht auf dem Beispiel in der obigen Beschreibung zum Erstellen von Dokumentations- und Applikatonsinformationen.

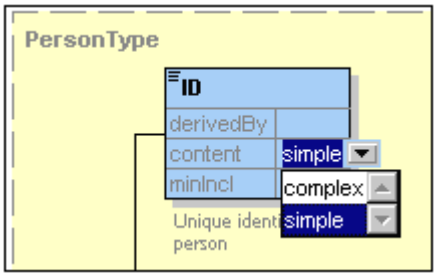
```
<xs:element name="session_date" type="xs:dateTime" nillable="true">
   <xs:annotation>
      <xs:documentation>Datum und Zeit des
Interviews</xs:documentation>
      <xs:appinfo
source="http://www.altova.com/datehandlers/interviews">separator =
:</xs:appinfo>
   </xs:annotation>
</xs:element>
```

Konfigurieren der Content Model-Ansicht
Sie können die Content Model-Ansicht für das gesamte Schema im Dialogfeld "Schema-Ansicht konfigurieren" (**Schema-Design | Ansicht konfigurieren**) konfigurieren. Nähere Informationen zu Konfigurationsoptionen finden Sie in der Benutzerreferenz im Abschnitt Ansicht konfigurieren. Beachten Sie, dass die hier definierten Einstellungen für das gesamte Schema, die Erzeugung der Schema-Dokumentation und die Druckausgabe gelten.

Ändern von Komponenteneigenschaften direkt im Content Model
Wenn die Content Model-Ansicht so konfiguriert ist, dass Komponenten mit zusätzlichen beschreibenden Eigenschaftszeilen im Komponentenkästchen angezeigt werden, können Sie diese Informationen bearbeiten und dadurch die Eigenschaften der Komponente ändern. Sie können diese Eigenschaftszeilen durch Klicken auf das Symbolleistensymbol "Vordefinierte Details hinzufügen" ein- und ausblenden. Sie können also zwischen einer Ansicht, die die definierten Eigenschaften enthält und einer Ansicht, die diese nicht enthält hin- und herschalten. So bearbeiten Sie Komponenteneigenschaften:

- Doppelklicken Sie auf das gewünschte Informationsfeld der jeweiligen Komponente und beginnen Sie mit der Eingabe oder Bearbeitung der Daten. Wenn eine vordefinierte Option zur Auswahl steht, erscheint eine Dropdown-Liste, aus der Sie den gewünschten Eintrag auswählen können. Geben Sie andernfalls den gewünschten Wert einfach ein.

© 2005 Altova GmbH



- Bestätigen Sie Ihre Eingabe durch Drücken der **Eingabetaste**. Die Details-Eingabehilfe wird aktualisiert, um Ihre Änderungen anzuzeigen.

Alternativ dazu können Sie die Eigenschaften einer Komponente in der Details-Eingabehilfe ändern. Die Änderungen werden in den Platzhalterfeldern übernommen - falls diese Anzeige aktiviert ist.

Dokumentieren des Content Models
Sie können detaillierte Dokumentation zu Ihrem Schema im HTML- und MS Word-Format erstellen. Für jede globale Komponente wird detaillierte Dokumentation generiert. Die Liste der globalen Komponenten wird im Inhaltsverzeichnis angezeigt, über das Sie Verknüpfungen zu den Content Models der einzelnen Komponenten herstellen können. Außerdem werden verwandte Elemente (wie z.B. Child-Elemente, complexTypes usw.) durch Hyperlinks referenziert, sodass Sie von einem Element zum anderen navigieren können. Um Schema-Dokumentation zu generieren, wählen Sie den Menübefehl **Schema-Design | Schema-Dokumentation generieren**.

Attribute und Identity Constraints
Attribute und Identity Constraints können in einem Fenster unterhalb der Content Model-Ansicht oder in Kästchen im Content Model selbst angezeigt werden. Durch Klicken auf die Symbole können Sie zwischen diesen beiden Ansichten wechseln. Eine Beschreibung zum Einfügen und Bearbeiten von Attributen und Identity Constraints finden Sie unter Definieren von Attributen für eine Komponente und Definieren von Identity Constraints für eine Komponente.

5.1.3 Arbeiten mit SchemaAgent

XMLSpy 2005 kann für die Verwendung mit Altova SchemaAgent konfiguriert werden.

SchemaAgent Technologie
SchemaAgent gestattet Benutzern, Beziehungen zwischen mehreren Schemas zu erstellen und zu bearbeiten. SchemaAgent besteht aus:

- Einem SchemaAgent Server, der Informationen über die Beziehungen zwischen Schemas in einem oder mehreren von Ihnen definierten Suchpfaden (Ordnern auf dem Netzwerk) enthält.
- Einem SchemaAgent Client, der anhand der Schema-Informationen vom SchemaAgent Server, mit dem er verbunden ist, (i) Beziehungen zwischen diesen Schemas erstellt; (ii) diese Schemas verwaltet (Umbenennen, Verschieben, Löschen von Schemas usw.); und, (iii) wenn XMLSpy 2005 als SchemaAgent Client verwendet wird, unter Verwendung von Informationen über andere Schemas, die er vom SchemaAgent Server abrufen kann, zum direkten Editieren von Schemas verwendet wird.

In dieser Konfiguration fungiert XMLSpy 2005 als SchemaAgent Client und muss mit einem SchemaAgent Server verbunden sein. Wenn XMLSpy 2005 für die Arbeit mit SchemaAgent konfiguriert wurde, enthalten die Eingabehilfen in der Schema-Ansicht nicht nur Komponenten aus dem derzeit in der Schema-Ansicht gerade aktiven Schema sondern auch Komponenten aus anderen Schemas im Netzwerk. Auf diese Art haben Sie direkten Zugriff auf diese Komponenten. Sie können das Content Model einer zu einem anderen Schema gehörenden Komponente in der Schema-Ansicht anzeigen und sie mit oder ohne Änderungen wiederverwenden. Außerdem können Sie Beziehungen zwischen Schemas erstellen und dadurch komplexe Schemas direkt in XMLSpy 2005 in Module aufteilen und verwalten.

Einrichten von XMLSpy 2005 als SchemaAgent Client
Um XMLSpy 2005 für die Arbeit als SchemaAgent Client zu konfigurieren, gehen Sie folgendermaßen vor:

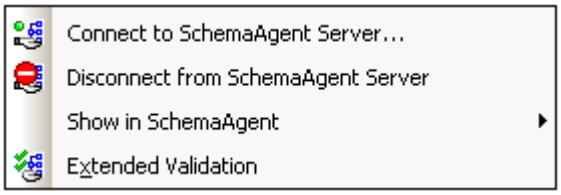
- Downloaden Sie SchemaAgent Server von der Altova Website und installieren Sie ihn auf Ihrem Netzwerk. SchemaAgent Server ist kostenlos erhältlich.
- Downloaden Sie Altova SchemaAgent 2005 (wird in dieser Dokumentation als SchemaAgent Client bezeichnet) von der Altova Website und installieren Sie das Programm auf demselben Rechner wie XMLSpy 2005. Für SchemaAgent Client muss eine Lizenz von Altova erworben werden. Die Altova Enterprise XML Suite und die Professional XML Suite enthalten beide den SchemaAgent Client sowie eine Lizenz dafür. (Die SchemaAgent Server Anwendung ist allerdings in diesen Paketen nicht enthalten.)
- Definieren Sie den/die Suchpfad(e) für den SchemaAgent Server. Eine ausführliche Erläuterung dazu finden Sie in der SchemaAgent Server-Dokumentation.
- Stellen Sie eine Verbindung zwischen XMLSpy 2005 und dem SchemaAgent Server her.

Hinweis:

- Damit XMLSpy 2005 als SchemaAgentClient arbeiten kann, muss **sowohl** SchemaAgent Server **als auch** SchemaAgent Client installiert sein.
- SchemaAgent Server und SchemaAgent Client müssen separat von der Altova Website heruntergeladen und installiert werden.
- Sobald SchemaAgent Server und SchemaAgent Client ordnungsgemäß installiert sind, stellt XMLSpy 2005 automatisch die notwendigen Verknüpfungen her.

Wichtig: Alle SchemaAgent-Produkte und alle mit SchemaAgent in Zusammenhang stehenden Produkte von Altova (einschließlich XMLSpy 2005) beginnend mit Version 2005 Release 3 sind **nicht** mit früheren Versionen von SchemaAgent oder mit SchemaAgent in Zusammenhang stehenden Produkten kompatibel.

SchemaAgent-Befehle in XMLSpy 2005
Die SchemaAgent-Funktion in XMLSpy 2005 steht nur in der Schema-Ansicht zur Verfügung und wird über die Menübefehle im Schema-Design-Menü *(siehe Abbildung unten),* die Symbolleiste und die Eingabehilfen in der Schema-Ansicht aufgerufen.



Über die Menübefehle stehen Ihnen allgemeine Verwaltungsfunktionen zur Verfügung. Die Eingabehilfen (und die Standardfunktion der Benutzeroberfläche wie z.B. Drag & Drop) dienen

 © 2005 Altova GmbH

zum Bearbeiten von Schemas.

In diesem Abschnitt wird beschrieben (i) wie man SchemaAgent Server und SchemaAgent Client installiert und (ii) wie man die in der Schema-Ansicht zur Verfügung stehenden SchemaAgent-Funktionen verwendet.

Installation von SchemaAgent Server

SchemaAgent Server muss für die Installation von der Altova Website heruntergeladen werden. Starten Sie die Installation anschließend, indem Sie auf die SchemaAgent Server-Installationsdatei klicken. Befolgen Sie die Anweisungen auf dem Bildschirm. Während der Installation können Sie zwischen mehreren Installationsoptionen wählen (*siehe Abbildung unten*). Mit Hilfe dieser Optionen können Sie bestimmen, wie viel Input vom Benutzer erfolgen muss, um SchemaAgent Server auf dem eigenen Rechner zu starten. Im Folgenden sind diese Optionen beschrieben:

Installation und Start von SchemaAgent Server als Standalone Server
Wenn SchemaAgent Server als Standalone Server installiert wird, kann er über das Menü **Start | Programme** oder ein Desktop-Symbol gestartet werden. Die Applikation wird als grafische Benutzeroberfläche geöffnet und nach dem Öffnen wird in der Task-Leiste ein Applikationssymbol angezeigt. SchemaAgent Server kann direkt in der GUI neu konfiguriert werden. Diese Installation eignet sich bei Installation auf einem lokalen Rechner.

Installation von SchemaAgent Server als interaktiver Dienst
Wenn Sie SchemaAgent Server als interaktiven Dienst installieren, wird SchemaAgent Server automatisch bei jedem Systemstart gestartet. Das SchemaAgent Server Applikatonssymbol erscheint in der Task-Leiste und die Applikation kann über dieses Symbol gestartet und geschlossen werden. Außerdem kann SchemaAgent Server auch mit Hilfe von

Betriebssystemtools (in Windows XP, **Verwaltung | Dienste**) gestartet und geschlossen werden. Wenn Sie SchemaAgent Server als interaktiven Dienst installieren, ersparen Sie sich das manuelle Starten und Schließen der Applikation. Wenn Sie in der Task-Leiste auf das Applikationssymbol klicken, können Sie die GUI ein- und ausblenden, die Konfiguration ändern und SchemaAgent Server schließen.
Diese Installationsart eignet sich vor allem, wenn SchemaAgent Server als Netzwerk-Server für mehrere Clients verwendet wird.

Installation von SchemaAgent Server als (nicht interaktiver) Dienst
Wenn Sie SchemaAgent Server als nicht interaktiven Dienst installieren, bleibt er für den Benutzer völlig unsichtbar. Das Betriebssystem startet SchemaAgent Server automatisch bei jedem Systemstart. In diesem Installationsmodus gibt es kein Symbol in der Task-Leiste. Um SchemaAgent Server zu konfigurieren (d.h. um Suchpfade zu definieren), müssen Sie zuerst SchemaAgent Server als interaktiven Dienst installieren, konfigurieren und ihn anschließend als nicht-interaktiven Dienst starten.

Installation von SchemaAgent Client

Um SchemaAgent Client (das Altova-Produkt **SchemaAgent™ 2005**) zu installieren, gehen Sie folgendermaßen vor:

1. Laden Sie das SchemaAgent Client Installationspaket von der Altova Website herunter. Wenn Sie die Altova Enterprise XML Suite oder die Professional XML Suite erworben haben, ist SchemaAgent Client darin inkludiert.
2. Starten Sie den Installationsvorgang (durch Doppelklick auf die Installationsdatei).
3. Geben Sie die Lizenzinformationen ein und speichern Sie sie, um das Produkt zu aktivieren. Die Lizenz für SchemaAgent Client muss im Altova-Online-Shop auf der Altova-Website erworben werden. Wenn Sie die Altova Enterprise XML Suite oder die Professional XML Suite erworben haben, ist die Lizenz für SchemaAgent Client bereits in diesem Produktpaket enthalten.

Starten der Installation
Um die Installation zu starten, doppelklicken Sie auf die Installationsdatei, die Sie heruntergeladen haben und befolgen Sie die Anweisungen des Installationsassistenten. Während der Installation werden Sie gefragt, ob Sie eine Verbindung zu einem SchemaAgent Server auf Ihrem Netzwerk herstellen möchten (falls vorhanden). Hier können Sie den SchemaAgent Server auswählen, zu dem dieser SchemaAgent Client die Verbindung herstellen soll. Sie müssen zu diesem Zeitpunkt keinen SchemaAgent Server definieren, sondern können diesen jedesmal, wenn Sie eine Verbindung zu einem SchemaAgent Server herstellen möchten, auswählen (Menüoption **Schema-Design | Verbindung zu SchemaAgent Server herstellen**).

Starten von SchemaAgent Client
Nachdem Sie SchemaAgent Client installiert haben, wird die Applikation in der Liste der Programme (**Start | Programme**) angezeigt. Um SchemaAgent Client zu starten, klicken Sie in der Liste der Programme auf den Eintrag SchemaAgent oder wählen Sie das SchemaAgent-Symbol in der Schnellstartleiste aus. Wenn Ihre Lizenz für SchemaAgent noch nicht registriert wurde, werden Sie beim Start von SchemaAgent aufgefordert, diese Informationen einzugeben.

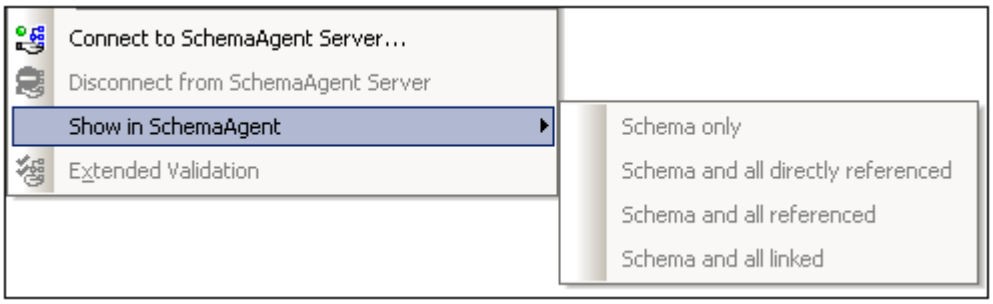
Hinweis: Informationen zur Verwendung von SchemaAgent Client finden Sie in der Dokumentation zu SchemaAgent Client.

© 2005 Altova GmbH

Herstellen einer Verbindung mit SchemaAgent Server

Nach Installation von SchemaAgent Server und SchemaAgent Client und Eingabe eines gültigen Lizenzschlüssels für SchemaAgent Client müssen Sie eine Verbindung zu SchemaAgent Server herstellen, damit XMLSpy 2005 SchemaAgent als SchemaAgent Client verwenden kann.

Bevor Sie eine Verbindung zu SchemaAgent Server hergestellt haben, ist im Menü **Schema-Design** nur der Befehl **Verbindung zu SchemaAgent Server herstellen,** aktiv. Alle anderen SchemaAgent-Befehle im Menü **Schema-Design** sind deaktiviert (*siehe Abbildung unten*). Die anderen Menüeinträge werden aktiv, sobald eine Verbindung zu SchemaAgent Server besteht.

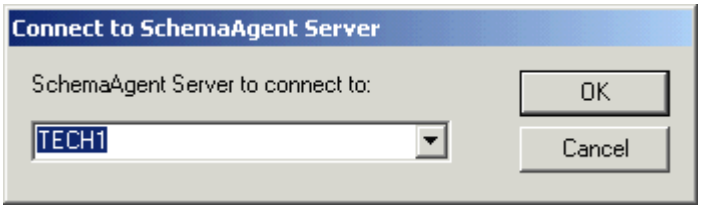


Hinweis: Wenn SchemaAgent Client nicht installiert ist, kann keine Verbindung zu SchemaAgent Server hergestellt werden.

Schritte beim Herstellen einer Verbindung

So stellen Sie eine Verbindung zu SchemaAgent Server her:

1. Es muss ein Schema im Suchpfad von SchemaAgent Server angezeigt werden, während es in der Schema-Design-Ansicht geöffnet ist. Wählen Sie nun **Schema-Design | Verbindung zu SchemaAgent Server herstellen** oder klicken Sie auf das gleichnamige Symbol . (Informationen zum Definieren eines Suchpfads finden Sie in der Dokumentation zu SchemaAgent Server.)
2. Klicken Sie in der Dropdown-Liste auf den "Nach unten"-Pfeil, um eine Liste der SchemaAgent Server anzuzeigen, die zur Verfügung stehen (*siehe Abbildung unten*).



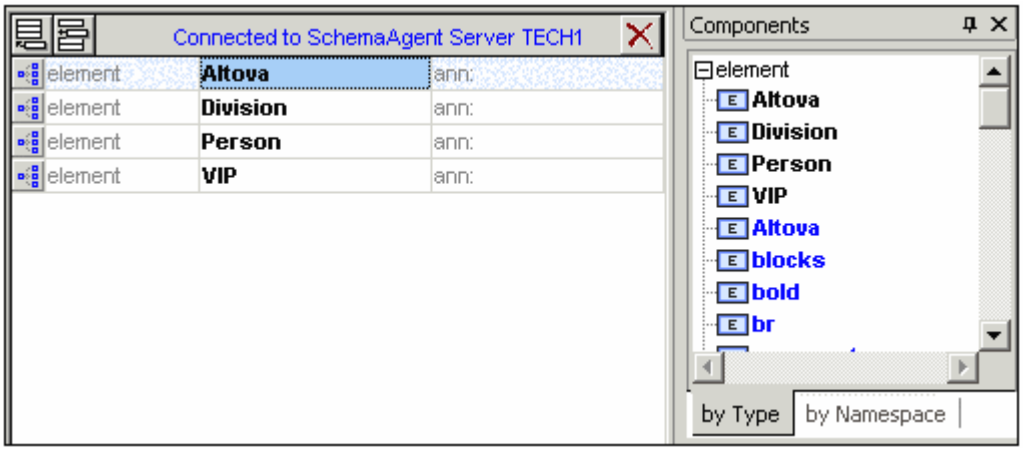
Hinweis zu Servern, die unter Windows XP SP2 laufen
Wenn der Name des SchemaAgent Servers im Dialogfeld "Verbindung zu SchemaAgent Server herstellen" aufgelistet ist, Sie jedoch keine Verbindung herstellen können, liegt dies möglicherweise daran, dass Ihr Server nicht an der Namensauflösung in Ihrem Netzwerk teilnimmt. Die Namensauflösung wird durch die Standardeinstellungen der Windows XP SP2 Firewall verhindert.

Um die Verbindung zu einem derartigen Server herzustellen, haben Sie folgende Möglichkeiten:

- Ändern Sie die Servereinstellungen, um die Namensauflösung zuzulassen oder
- Geben Sie im Bearbeitungsfeld des Dialogfelds zur Verbindungsherstellung die IP-Adresse des Servers ein.

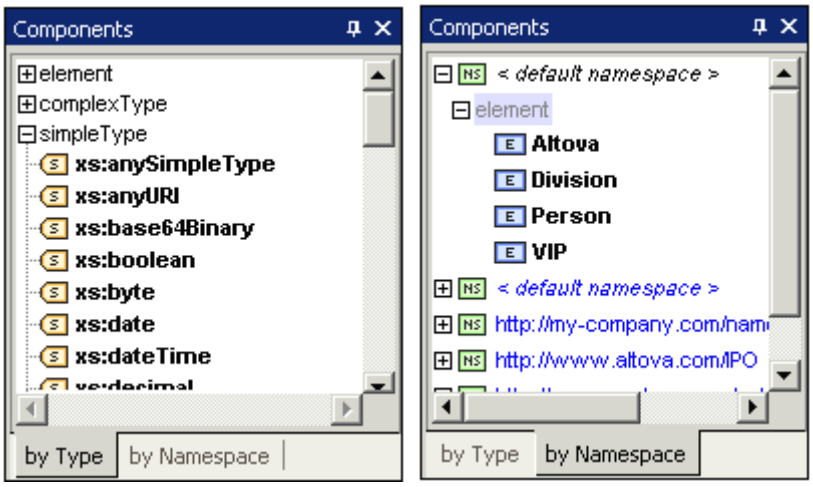
Sie müssen diese Einstellungen nur einmal vornehmen, da SchemaAgent Client den Connection String der letzten funktionierenden Verbindung speichert.

3. Wählen Sie den gewünschten Server aus und klicken Sie auf OK. XMLSpy 2005 stellt nun eine Verbindung zum gewünschten SchemaAgent Server her. Beachten Sie: Wenn SchemaAgent Client nicht installiert ist, kann keine Verbindung zum SchemaAgent Server hergestellt werden.



In der Schema/WSDL-Ansicht werden nun in der Ansicht der globalen Elemente der Text "Verbunden mit SchemaAgent Server ..." sowie die entsprechenden Symbole in der Symbolleiste angezeigt. Sie haben nun Zugriff auf alle Schemas und Schema-Konstrukte, die unter dem Server-Suchpfad zur Verfügung stehen. SchemaAgent Schema-Konstrukte: Globale Elemente, complexTypes, simpleTypes usw. werden unterhalb der Konstrukte des aktiven Schemas blau und fett angezeigt. Die Schema-Konstrukte des aktiven Schemas erscheinen in Schwarz, während diejenigen, die über SchemaAgent verfügbar sind, blau sind.

© 2005 Altova GmbH

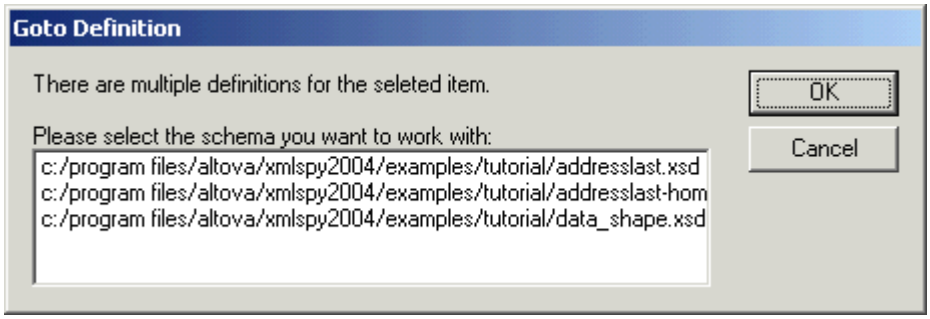


Schema-Konstrukte können in der Schema/WSDL-Ansicht nach Typ oder nach Namespace (Register "by Type" und "by Namespace") angezeigt werden.

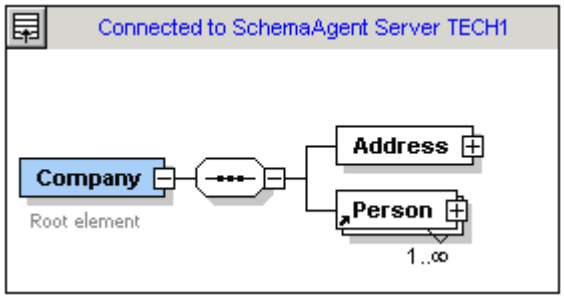
Öffnen von Schemas aus dem Suchpfad

In diesem Beispiel wird gezeigt, wie man ein Schema öffnet, das unter einem in SchemaAgent Server definierten Suchpfad gespeichert ist. Als aktives Schema wird die Datei `DB2schema.xsd` aus dem Ordner `..\Tutorial` verwendet. Das Register `By Type` der Komponenten-Eingabehilfe ist das aktive Register.

1. Scrollen Sie in der Komponenten-Eingabehilfe hinunter zum blauen Eintrag `Company` in der Komponente "Element" und doppelklicken Sie darauf. Daraufhin wird das Dialogfeld "Gehe zu Definition" geöffnet.



2. Klicken Sie auf den Eintrag `Addresslast.xsd` und bestätigen Sie mit OK. Das Schema `addresslast.xsd` wird geöffnet und das Content Model des Elements `Company wird angezeigt.`



Hinweis: Wenn Sie auf ein SchemaAgent Schema-Konstrukt, wie z.B. ein Element, einen complexType oder einen simpleType doppelklicken, wird das dazugehörige Schema (sowie alle anderen inkludierten Schemas) in XMLSpy 2005 geöffnet.

Verwenden von Schema-Konstrukten

XML-Schema bietet drei grundlegende Methoden, um Schemas in Module aufzugliedern: Import, Include und Redefine (IIR). Bei jeder Methode gelten andere Namespace-Anforderungen. Diese werden jedoch automatisch beim Erstellen von IIRs von SchemaAgent Client und XMLSpy 2005 überprüft.

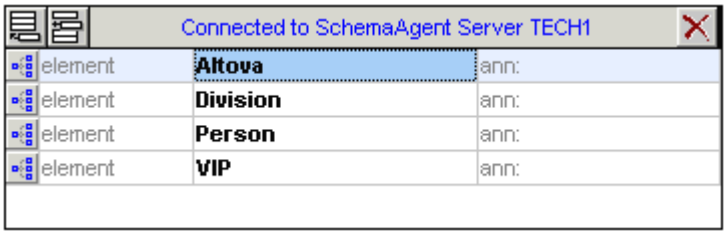
Import-, Include- und Redefine-Anweisungen (IIR)
Es gibt unterschiedliche Arten, um ein Schema-Konstrukt "einzufügen":

- Sie können globale Elemente direkt aus der Komponenten-Eingabehilfe (in der Schema/WSDL Design-Ansicht) in das Content Model einer Schemakomponente ziehen
- Komponenten, wie z.B. complexTypes und simpleTypes können aus einem Listenfeld ausgewählt werden, das automatisch geöffnet wird, wenn Sie ein neues Element/Attribut usw. definieren.
- Komponenten wie z.B. complexTypes können bei der Erstellung/Aktualisierung dieser Konstruktart in der Details-Eingabehilfe ausgewählt werden.

Integrieren von Schema-Komponenten
In diesem Beispiel wird die Datei `DB2schema.xsd` aus dem Ordner `..\Tutorial` als aktives Schema verwendet; das Register `By Type` der Komponenten-Eingabehilfe ist aktiv.

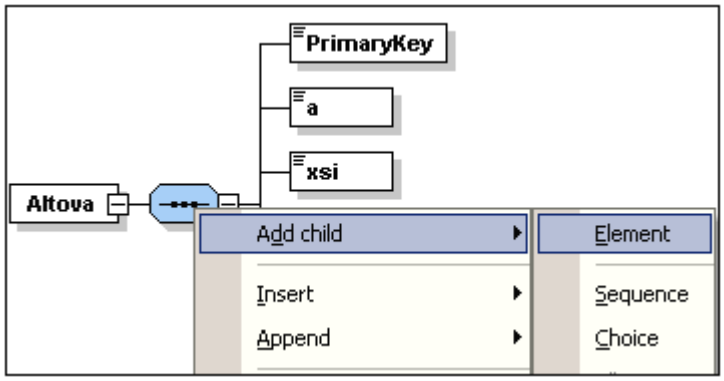
So verwenden Sie Schema-Konstrukte aus SchemaAgent Server Schemas:

1. Öffnen Sie die Datei `DB2Schema.xsd` und benennen Sie sie für dieses Beispiel in `Altova-office` um.

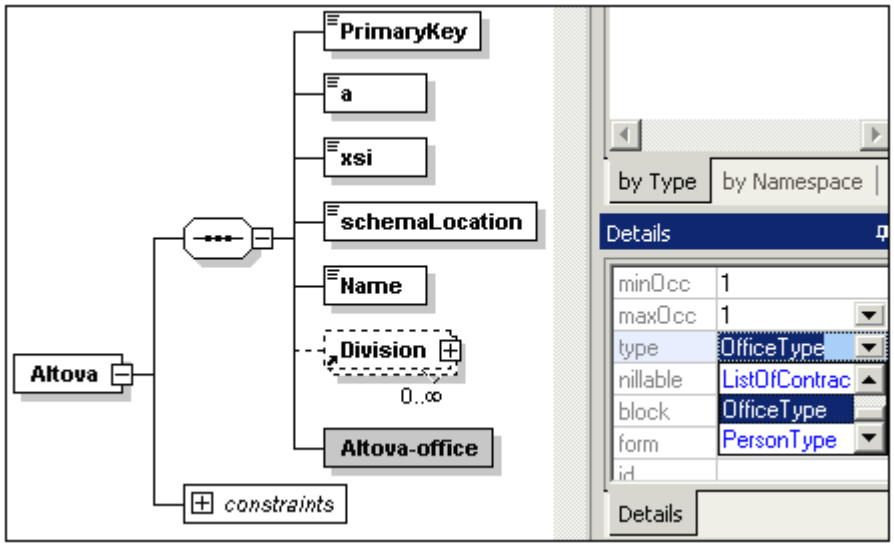


2. Klicken Sie in der Schema-Übersicht auf das Element `Altova,` um dessen Content Model aufzurufen.
3. Rechtsklicken Sie auf den `Altova` sequence-Kompositor und wählen Sie die Menüoption **Child hinzufügen | Element**. Beachten Sie, dass nun automatisch ein Listenfeld mit allen globalen Elementen aus dem Serverpfad geöffnet wird. Wenn Sie eines auswählen, wird dieses Element integriert.

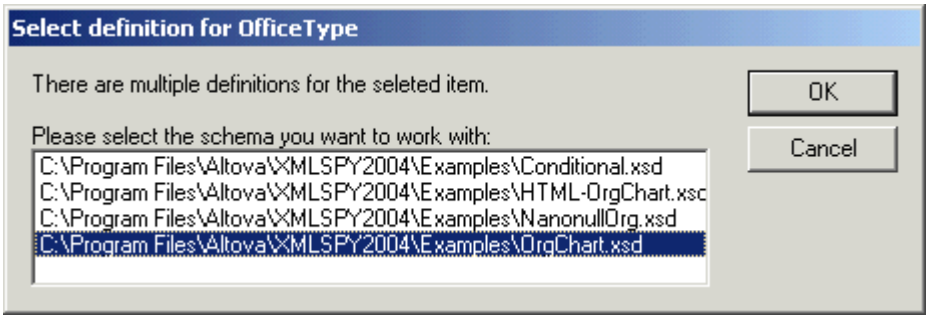
© 2005 Altova GmbH



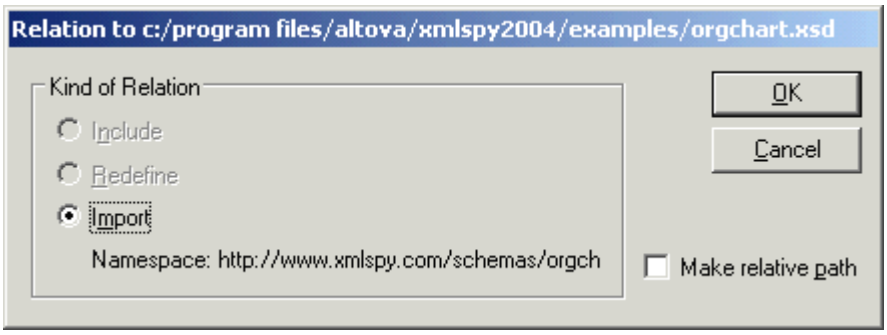
4. Geben Sie `Altova-office` als Namen für dieses neue Element ein und drücken Sie die Eingabetaste.
5. Klicken Sie in der Details-Eingabehilfe auf das Listenfeld `type` und wählen Sie den Eintrag `OfficeType` aus.



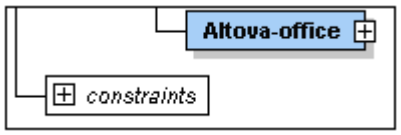
Daraufhin wird das Dialogfeld "Definition für OfficeType auswählen" geöffnet.



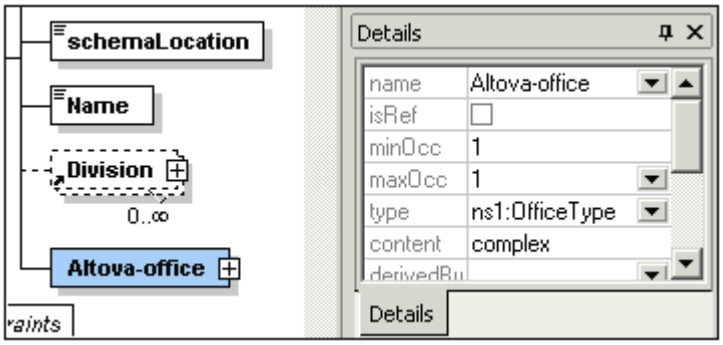
6. Wählen Sie `Orgchart.xsd` und klicken Sie auf OK.



7. Klicken Sie im nächsten Dialogfeld auf OK. Der Befehl "Import" wurde automatisch ausgewählt. Das Element `Altova-office` weist nun ein Plus-Zeichen zum Erweitern auf.

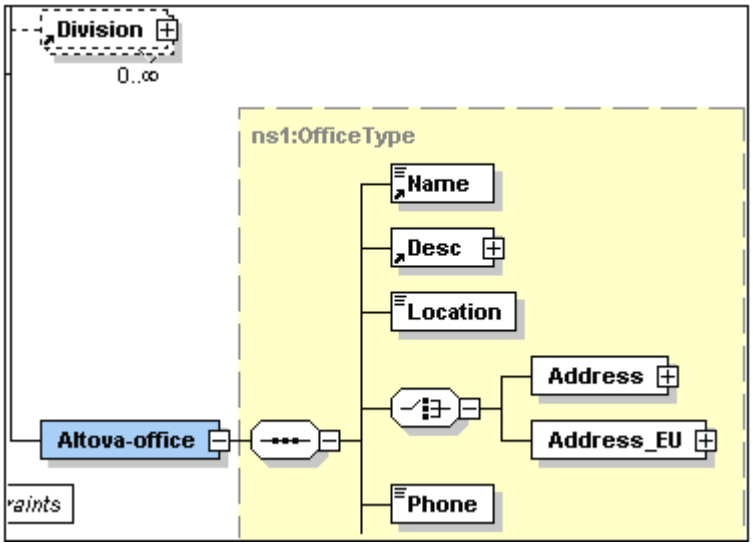


Bitte beachten Sie, dass sich der Eintrag `type` entry in der Details-Eingabehilfe geändert hat; er wird nun als `ns1:OfficeType` angezeigt, da die Schemadatei `Orgchart.xsd` importiert wurde und die beiden Schemas unterschiedliche Target Namespaces haben müssen. Außerdem wurde zum Schema ein Import-Befehl hinzugefügt.



8. Klicken Sie auf das Erweiterungssymbol, um das Content Model von `OfficeType` anzuzeigen.

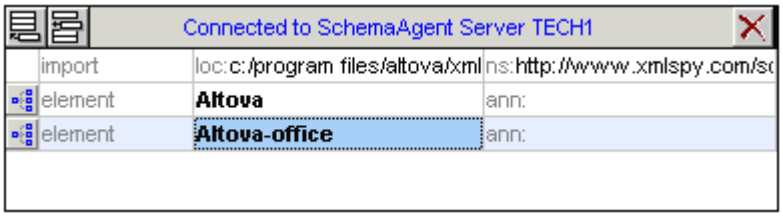
 © 2005 Altova GmbH



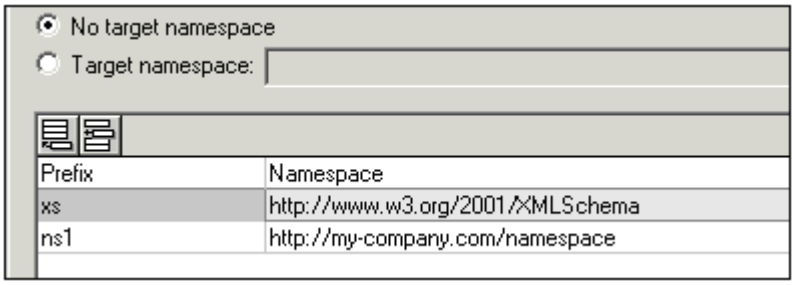
9. Drücken Sie F8, um das Schema zu validieren. Daraufhin sollte die Meldung, dass das Schema gültig ist, erscheinen.

Aufräumen des Schemas:

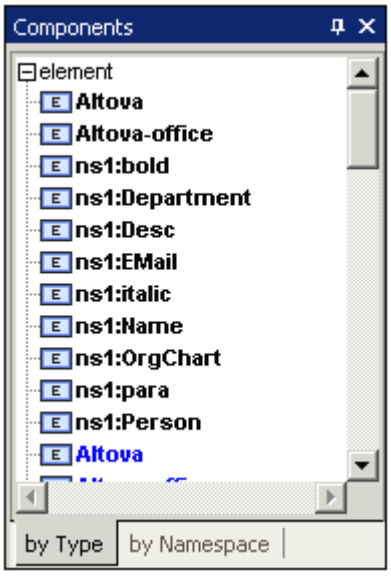
1. Löschen Sie das Element `Division` im Content Model.
2. Klicken Sie auf das Symbol zum Anzeigen der globalen Elemente, um in die Schema-Übersicht zu wechseln.
3. Löschen Sie die folgenden globalen Elemente: `Division`, `Person` und `VIP`.



4. Wählen Sie die Menüoption **Schema-Design | Schema-Einstellungen**, um zu sehen, inwiefern sich die Namespace-Einstellungen geändert haben.



Das `ns1`-Präfix wurde automatisch zu `my-company.com/namespace` hinzugefügt. Die Komponente- und die Details-Eingabehilfe zeigen alle importierten Konstrukte mit dem ns1: namespace Präfix an.

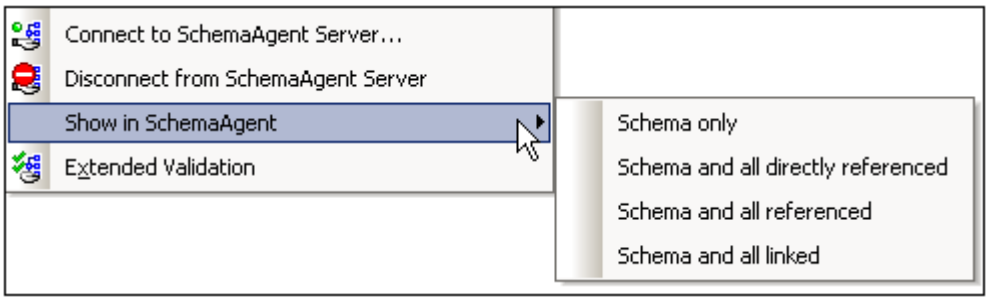


Hinweis:

- Änderungen, die in XMLSpy 2005 unter Verwendung von SchemaAgent vorgenommen wurden, werden automatisch in allen anderen Schemas im SchemaAgent Serverpfad, der das geänderte Schema referenziert, übernommen.
- Unter Umständen werden Konstrukte wie Elemente, simpleTypes usw. in den Eingabenhilfen doppelt angezeigt (in Schwarz und Blau), wenn sich das Schema, mit dem Sie gerade arbeiten, auch unter dem SchemaAgent Serverpfad befindet.

Anzeigen von Schemas in SchemaAgent

Um mit dem aktiven Schema und den damit verknüpften Schemas in SchemaAgent zu arbeiten, klicken Sie auf die Menüoption **Schema-Design | In SchemaAgent anzeigen | *Schema und alle damit verbundene*** *(siehe Abbildung unten).*



Sie haben die Option, entweder nur das aktive Schema in SchemaAgent zu öffnen oder das aktive Schema zusammen mit entweder (i) allen direkt referenzierten Schemas oder (ii) allen referenzierten Schemas oder (iii) allen verbundenen Schemas zu öffnen.

Erweiterte Validierung

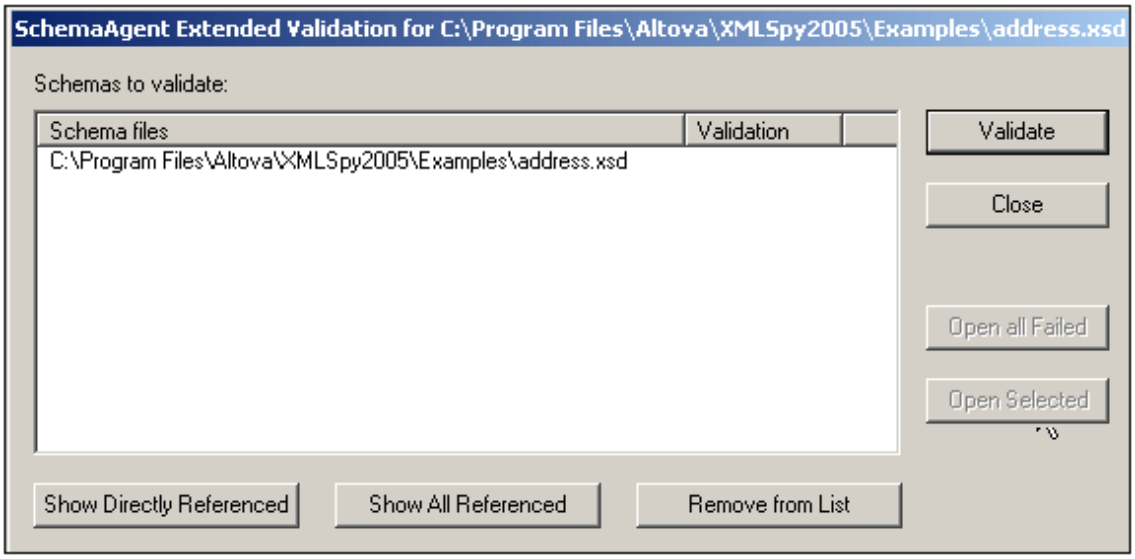
XMLSpy 2005 gestattet Ihnen in Verbindung mit SchemaAgent nicht nur das aktive Schema zu validieren, sondern auch Schemas, die mit dem aktiven Schema in Zusammenhang stehen.

© 2005 Altova GmbH

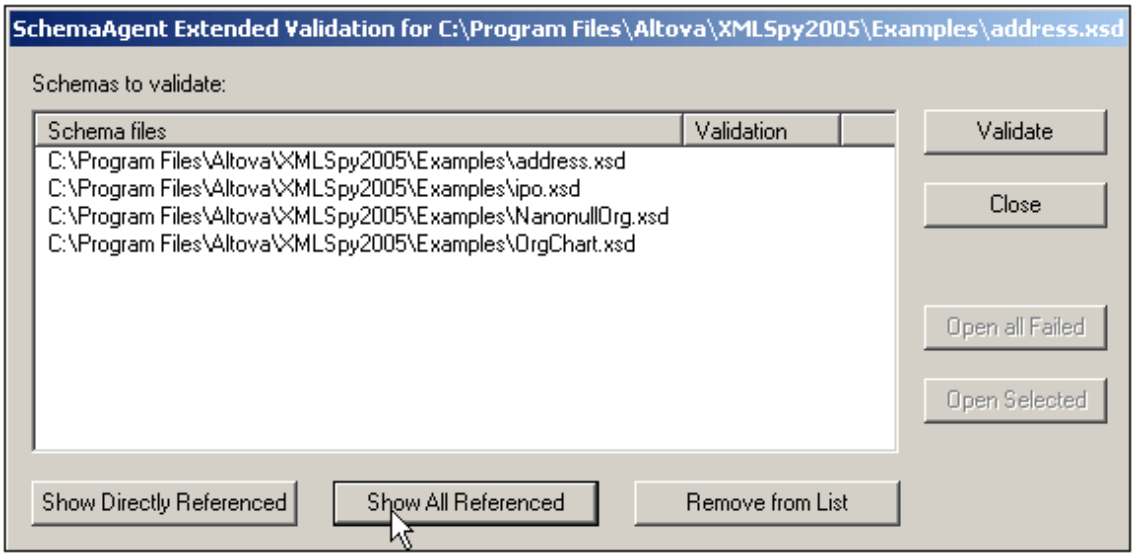
SchemaAgent unterscheidet bei der erweiterten Validierung zwischen: (i) direkt referenzierten Schemas und (ii) referenzierten Schemas.

Die Funktionsweise der erweiterten Validierung wird im Folgenden anhand eines Beispiels beschrieben. In diesem Beispiel wird als aktive Schemadatei in der Schema/WSDL-Ansicht von XMLSpy 2005 `address.xsd` verwendet. Gehen Sie bei der erweiterten Validierung vor wie folgt:

1. Klicken Sie in der Symbolleiste auf das Symbol "Erweiterte Validierung" oder wählen Sie den Menüeintrag **Schema-Design | Erweiterte Validierung**. Daraufhin wird das Dialogfeld "Erweiterte Validierung" geöffnet. Hier können Sie angeben, ob nur das aktive Schema oder auch ein oder mehrere damit verknüpfte Schemas validiert werden sollen.

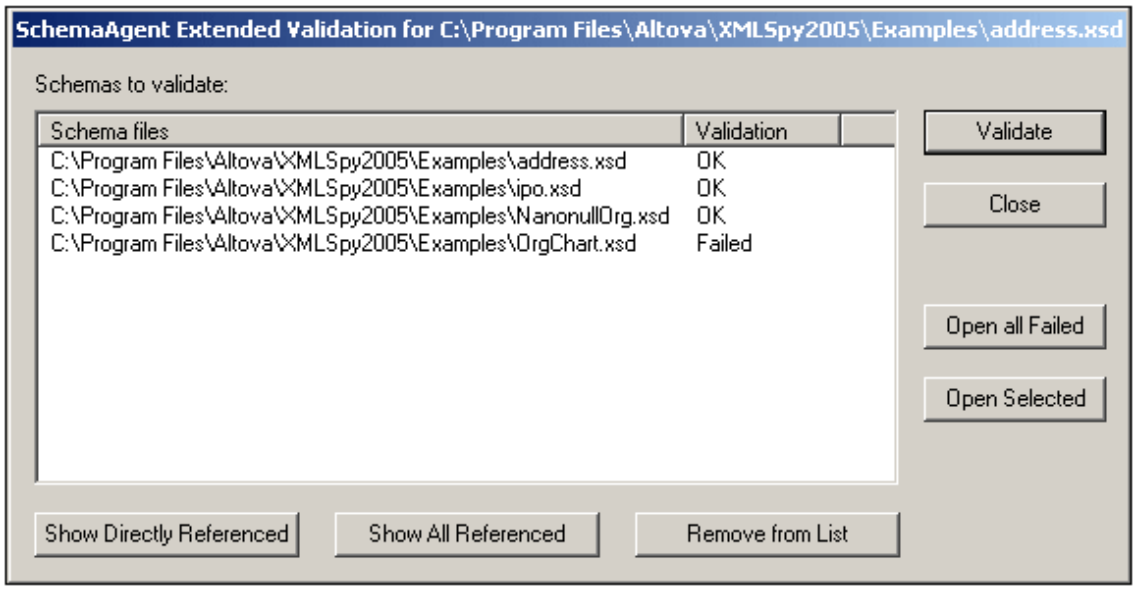


2. Um Schemas zur Liste hinzuzufügen, klicken Sie je nach Bedarf entweder auf "Direkt referenzierte anzeigen" oder auf "Alle referenzierten anzeigen". In diesem Beispiel haben wir die Option "Alle referenzierten anzeigen" gewählt, woraufhin alle referenzierten Dateien in die Liste übernommen werden.



Sie können zu diesem Zeitpunkt ein Schema aus der Liste entfernen ("Aus Liste löschen'").

3. Klicken Sie auf die Schaltfläche "Validieren", um alle Schemas im Listenfeld zu validieren.



In der Spalte "Validieren" sehen Sie, ob die Validierung erfolgreich war oder fehlgeschlagen ist.

Sie können nun die Schemas, bei denen ein Validierungsfehler aufgetreten ist oder eine Gruppe von ausgewählten Schemas in XMLSpy 2005 öffnen.

© 2005 Altova GmbH

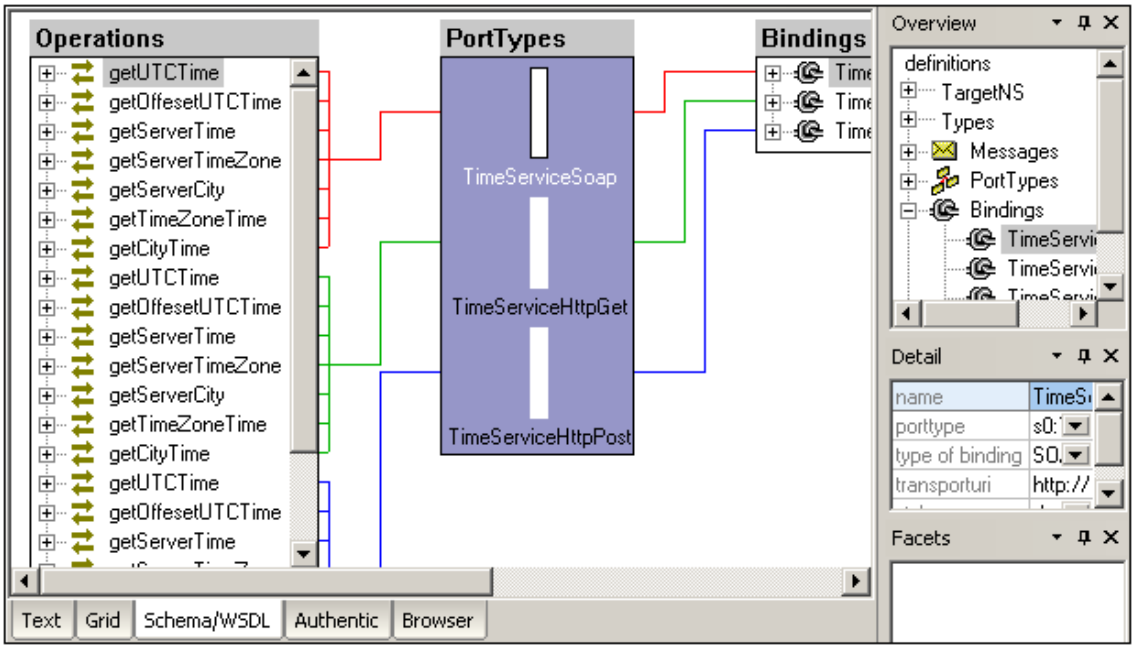
5.2 WSDL Design-Ansicht

In XMLSpy 2005 können Sie in der WSDL Design-Ansicht WSDL-Dokumente erstellen, anzeigen und bearbeiten. Die WSDL Design-Ansicht bietet die folgenden Funktionen:

- Grafische Anzeige aller WSDL-Elemente, gruppiert nach Operations, PortTypes, Bindings und Services sowie eine Anzeige der WSDL-Dateistruktur.
- Direkte Bearbeitung von WSDL-Elementen mittels Drag & Drop.
- die Möglichkeit, WSDL-Elemente, die in der grafischen Ansicht zu sehen sind, hinzuzufügen, anzuhängen und zu löschen (kontextsensitives Menü).
- die Möglichkeit, Werte in der Details-Eingabehilfe einzugeben und zu bearbeiten.
- WSDL-Validierung gegen einen W3C Working Draft.
- Automatisches Wechseln in die Schema Design-Ansicht, wenn sich ein referenziertes oder importiertes Schema als ungültig herausstellt.
- Bearbeiten von Schema-Typen in der WSDL Design-Ansicht.
- Generierung von WSDL-Dokumentation in MS Word oder HTML.

Die Benutzeroberfläche im Überblick
In der Abbildung unten sehen Sie eine in der WSDL Design-Ansicht geöffnete WSDL-Datei. Die Benutzeroberfläche besteht aus drei Hauptbereichen:

- dem Projekt- und dem Info-Fenster
- dem Hauptfenster
- den Eingabehilfen-Fenstern



Bitte beachten Sie, dass das Projekt- und das Info-Fenster in der obigen Abbildung geschlossen wurden, damit man mehr von der WSDL-Ansicht sieht. Um diese Fenster anzuzeigen, wählen Sie **Fenster | Projekt-Fenster** (oder **Info-Fenster**).

Projekt- und Info-Fenster

- Im Projekt-Fenster sehen Sie die Dateien eines Projekts. Damit können Sie ein Projekt besser verwalten. Nähere Informationen zum Arbeiten mit Projekten finden Sie in der Benutzerreferenz unter dem Kapitel Menü "Projekt".
- Im Info-Fenster sehen Sie nähere Informationen über das im Hauptfenster markierte

Element oder Attribut.

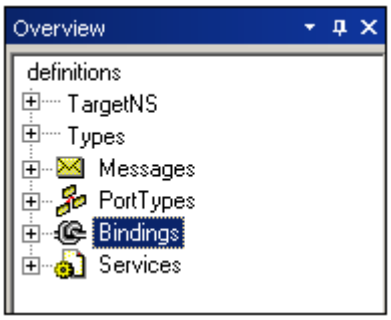
Hauptfenster
In diesem Bereich wird die WSDL-Datei bearbeitet. Beachten Sie, was die Dateiansicht betrifft, die folgenden Punkte:

- Wenn Sie eine WSDL-Datei öffnen, wird diese automatisch in der WSDL Design-Ansicht geöffnet.
- Sie können eine WSDL-Datei auch in der Text- und der Enhanced Grid-Ansicht anzeigen. Klicken Sie dazu auf das entsprechende Register.
- Wenn die WSDL-Datei eine Referenz zu einem XML-Schema enthält, können Sie dieses Schema durch Auswahl von **WSDL | Typen | Schema bearbeiten** öffnen und bearbeiten. Die Schemadatei wird in der Schema Design-Ansicht geöffnet.
- Wenn eine verknüpfte Schemadatei geöffnet ist, darf die Ansicht der WSDL-Datei nicht gewechselt werden (z.B. von der WSDL Design-Ansicht in die Textansicht). Stellen Sie vor dem Wechseln der Ansicht für die WSDL-Datei sicher, dass Sie die Änderungen an der Schemadatei gespeichert haben und die Datei geschlossen haben.

Nähere Informationen zum Operations-, PortTypes-, Bindings-, and Services-Fenster finden Sie unter WSDL Design-Komponenten.

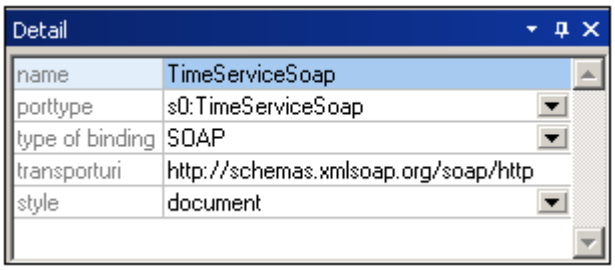
Eingabehilfen
Es gibt zwei Eingabehilfen: Das Übersichtsfenster und das Details-Fenster.

Übersichtsfenster



Das Übersichtsfenster enthält eine Übersicht über die Struktur des WSDL-Dokuments. Sie sehen darin die Komponenten der WSDL-Datei in Form einer Baumstruktur in der allgemeinen Reihenfolge, in der sie im Text vorkommen. Sie können die Komponenten erweitern und reduzieren, um die Sub-Komponenten ein- bzw. auszublenden. Wenn Sie im Übersichtsfenster eine Komponente auswählen, werden ihre Eigenschaften im Details-Fenster angezeigt, wo sie bearbeitet werden können.

Details-Fenster



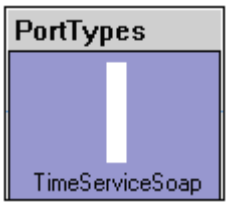
Im Details-Fenster werden Komponenten des WSDL-Dokuments sowie deren Eigenschaften einzeln angezeigt. Sie können diese Eigenschaften im Details-Fenster bearbeiten. Um eine Komponente (und ihre Eigenschaften) im Details-Fenster anzuzeigen, wählen Sie diese

 © 2005 Altova GmbH

entweder im Hauptfenster oder im Übersichtsfenster aus.

5.2.1 WSDL Design-Komponenten

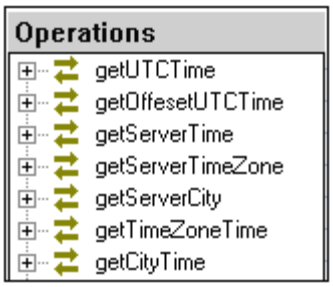
Das Fenster PortTypes



Im Fenster "PortTypes" werden alle PortTypes der WSDL-Datei angezeigt. Jeder PortType ist grafisch mit der Liste der Operationen (Operations) für den spezifischen Dienst (Service) verbunden. Wenn Sie in ein PortType-Fenster klicken, werden alle damit verknüpften Operationen im Fenster "Operations" angezeigt. Außerdem ist jeder PortType grafisch mit dem dazugehörigen Binding im Fenster "Bindings" verknüpft.

Wenn Sie mit der rechten Maustaste auf ein PortType-Rechteck klicken, wird ein kontextsensitives Menü geöffnet. Sie können einen PortType anhängen oder löschen oder eine Operation anhängen.

Das Fenster "Operations"



Im Fenster "Operations" werden alle Operationen für jeden PortType angezeigt. Die mit einem PortType verknüpften Operationen werden grafisch durch einen Konnektor (Verbindungslinie) zum Fenster "PortType" gekennzeichnet.

Wenn Sie auf das Erweiterungssymbol "+" einer Operation klicken, werden die Input- und Output-Messages angezeigt. Wenn eine Message aus mehreren Teilen besteht, werden die Parameter der einzelnen "Parts" angezeigt, wenn Sie auf das Erweiterungssymbol klicken. Bitte beachten Sie, dass diese Message Parts auch in der Details-Eingabehilfe angezeigt werden. Hier können Sie Parameter eingeben und direkt editieren.

Bei Rechtsklick in das Fenster "Operations" wird ein kontextsensitives Menü geöffnet. Sie können Operationen anhängen, einfügen oder löschen und Input- und Output-Messages zu einer bestimmten Operation hinzufügen.

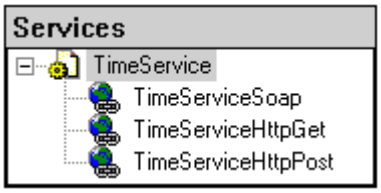
Bindings

Im Fenster "Bindings" werden die Bindings für jede der mit einem bestimmten PortType verknüpften Operationen angezeigt. In der Details-Eingabehilfe werden alle relevanten Parameter angezeigt:

- das `Name`-Attribut, das benutzerdefiniert ist
- das `(port)Type`-Attribut, das den jeweiligen PortType referenziert
- die Art des Binding: `SOAP`, `http/GET` oder `http/POST`
- das `Transport`-Attribut, das die zu verwendende `transporturi` angibt und das
- `Style`-Attribut, das rpc oder document sein kann
  Für jede Operation müssen die SOAP-Aktion sowie Eingabe- und Ausgabekodierung definiert sein.

Wenn Sie mit der rechten Maustaste auf einen Eintrag klicken, wird ein kontextsensitives Menü angezeigt. Bindings können angehängt oder gelöscht werden; es können Extensibility-Elemente zu Input- oder Output Messages hinzugefügt oder daraus gelöscht werden.

Services



Im Fenster "Services" werden die aktuell definierten Webservices angezeigt. In der Details-Eingabehilfe werden Name, Binding und Pfad des Dienstes angezeigt und Sie können diese Informationen darin direkt eingeben oder bearbeiten.

Durch Rechtsklick auf einen Eintrag wird ein Kontextmenü geöffnet. Services können angehängt oder gelöscht werden; Ports können an ein Service angehängt oder daraus gelöscht werden.

 © 2005 Altova GmbH

6 Authentic-Ansicht

Die Authentic-Ansicht (siehe Abbildung unten) ist eine grafische Darstellung Ihres XML-Dokuments. Sie können damit XML-Dokumente **ohne** Markup-Code und **mit** den entsprechenden Formatierungs- und Dateneingabefunktionen wie z.B. Eingabefeldern, Listenfeldern und Optionsfeldern darstellen. Die Daten, die der Benutzer in der Authentic-Ansicht eingibt, werden in die XML-Datei eingegeben.

In der Authentic-Ansicht lassen sich XML-Dokumente und Datenbanken (DB) über eine einfache Benutzeroberfläche problemlos editieren.

Um ein XML-Dokument in der Authentic-Ansicht anzeigen und bearbeiten zu können, muss es mit einem **StyleVision Power Stylesheet** verknüpft werden, das in Altova StyleVision 2005 erstellt wird. Ein StyleVision Power Stylesheet (`.sps`-Datei) ist im Grunde ein XSLT Stylesheet. Es definiert eine Ausgabedarstellung für ein XML-Dokument, das auch Dateneingabemechanismen enthält. Benutzer der Authentic-Ansicht können daher Daten zurück in die XML-Datei oder die DB schreiben. Ein StyleVision Power Stylesheet basiert auf einem Schema und ist schemaspezifisch. Wenn Sie ein StyleVision Power Stylesheet verwenden möchten, um eine XML-Datei in der Authentic-Ansicht zu editieren, muss es sich dabei um ein StyleVision Power Stylesheet handeln, das auf demselben Schema basiert, wie dasjenige, auf dem die XML-Datei basiert.

Verwendung der Authentic-Ansicht

- Wenn eine XML-Datei geöffnet ist, können Sie durch Klicken auf die Schaltfläche **Authentic** am unteren Rand des Dokumentenfensters in die Authentic-Ansicht wechseln. Wenn der XML-Datei noch kein StyleVision Power Stylesheet zugewiesen wurde, werden Sie aufgefordert, dies jetzt zu tun. Es muss ein StyleVision Power Stylesheet verwendet werden, das auf demselben Schema basiert, wie die XML-Datei.

- Wenn Sie den Befehl **Datei | Neu** wählen und anschließend auf die Schaltfläche "StyleVision Stylesheet auswählen..." klicken, wird in der Authentic-Ansicht eine neue XML-Datei erstellt und angezeigt. Diese neue Datei ist eine Vorlagendatei, die mit dem StyleVision Power Stylesheet verknüpft ist, das Sie öffnen. Sie kann bereits verschiedene Ausgangsdaten enthalten. Diese Ausgangsdaten befinden sich in einer XML-Datei (einer XML-Vorlagendatei), die optional mit dem StyleVision Power Stylesheet verknüpft werden kann. Nachdem die Authentic-Ansicht einer XML-Datei nun angezeigt wird, können Sie Daten darin eingeben und die Datei speichern.

Dieser Abschnitt enthält:

- eine Übersicht über die Benutzeroberfläche
- eine Beschreibung der Symbolleistensymbole der Authentic-Ansicht
- eine Beschreibung der Ansichtsmodi im Authentic-Ansicht-Hauptfenster
- eine Beschreibung der Eingabehilfen und deren Verwendung
- eine Beschreibung der verschiedenen in der Authentic-Ansicht verfügbaren Kontextmenüs
- eine Erläuterung zur Verwendung der verschiedenen Funktionen der Authentic-Ansicht

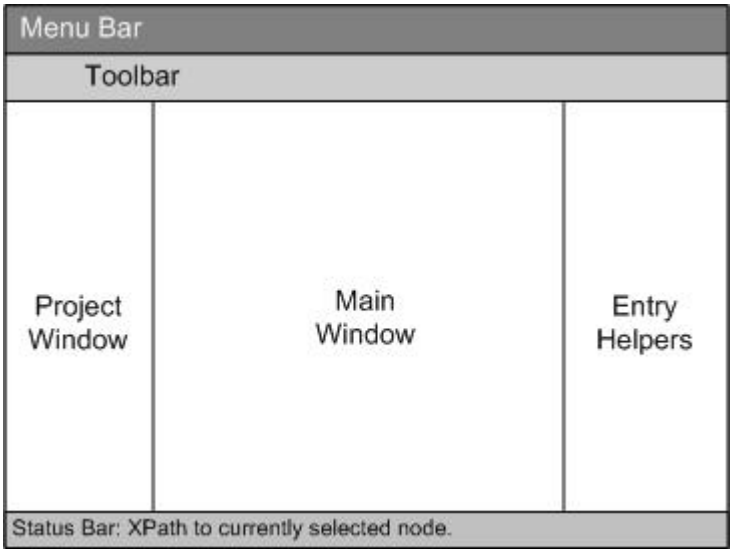
Weitere Informationen zur Authentic-Ansicht finden Sie:

- im Tutorial zur Authentic-Ansicht, in dem beschrieben wird, wie Sie mit der Authentic-Ansicht arbeiten. Sie finden das Tutorial in der Dokumentation zu Altova XMLSpy 2005 und zur Altova Authentic 2005 Desktop Edition (siehe Abschnitt Tutorials) sowie online.
- Eine ausführliche Beschreibung der Menübefehle der Authentic-Ansicht finden Sie im Abschnitt "Benutzerreferenz" Ihrer Produktdokumentation.

© 2005 Altova GmbH

6.1 Übersicht über die Benutzeroberfläche

Die Authentic-Ansicht besteht aus einer Menüleiste am oberen Rand des Fensters und drei Bereichen: dem Projekt-Fenster, dem Hauptfenster und den Eingabehilfen-Fenstern (siehe Abbildung unten).



Menüleiste
Die Menüs der Menüleiste werden in der Produktdokumentation im Abschnitt "Benutzerreferenz" näher beschrieben.

Symbolleiste
Eine Beschreibung der Schaltflächen der Symbolleiste finden Sie im Abschnitt Symbolleistenschaltflächen der Authentic-Ansicht.

Projekt-Fenster
XML-, XSL-, HTML-Schema und Entity-Dateien können in einem Projekt zusammen gruppiert werden. Verwenden Sie zum Erstellen und Bearbeiten der Liste der Projektdateien die Befehle im Menü "Projekt" (Beschreibung siehe Abschnitt "Benutzerreferenz" der Produktdokumentation). Die Liste der Projektdateien wird im Projekt-Fenster angezeigt. Um eine Datei im Projekt-Fenster aufzurufen, doppelklicken Sie darauf.

Hauptfenster
Das Hauptfenster ist der Bereich, in dem das XML-Dokument angezeigt und bearbeitet wird. Eine Beschreibung dazu finden Sie im Abschnitt Authentic-Ansicht Hauptfenster.

Eingabehilfen
Dieser Bereich enthält drei Eingabehilfen-Fenster: Elemente, Attribute und Entities. Welche Einträge in diesen Fenstern (Element- und Attribute-Eingabehilfen) zu sehen sind, ist vom Kontext abhängig, d.h. davon, an welcher Stelle im Dokument sich der Cursor gerade befindet. Um ein Element oder eine Entity in das Dokument einzugeben, doppelklicken Sie in der Eingabehilfe darauf. Der Wert eines Attributs wird in der Attribute-Eingabehilfe im Feld "Wert" des entsprechenden Attributs eingegeben. Nähere Informationen dazu finden Sie im Abschnitt Authentic-Ansicht Eingabehilfen.

Statusleiste
In der Statusleiste wird der XPath zum aktuell ausgewählten Node angezeigt.

Kontextmenüs
Dies sind Menüs, die angezeigt werden, wenn Sie im Hauptfenster auf die rechte Maustaste klicken. Es stehen Ihnen kontextsensitive Editierbefehle zur Verfügung, mit Hilfe derer Sie die Struktur und den Inhalt des ausgewählten Nodes bearbeiten können. Diese Befehle gestatten das Einfügen, Anhängen oder Löschen eines Nodes, das Hinzufügen von Entities und das Ausschneiden und Einfügen von Inhalt.

© 2005 Altova GmbH

6.2 Authentic-Ansicht Symbolleistenschaltflächen

Die Schaltflächen der Symbolleiste der Authentic-Ansicht sind Befehls-Shortcuts. Einige Symbole kennen Sie sicher bereits von anderen Windows-Anwendungen oder anderen Altova-Produkten her, andere sind vielleicht neu für Sie. In diesem Abschnitt werden die für die Authentic-Ansicht spezifischen Befehle beschrieben.

In der nachfolgenden Beschreibung sind verwandte Befehle gemeinsam gruppiert.

Wechseln in die Authentic-Ansicht

Wenn das XML-Dokument mit einem StyleVision Power Stylesheet verknüpft ist, wechseln Sie durch Auswahl des Befehls **Ansicht | Authentic** aus einer anderen Ansicht in die Authentic-Ansicht.
Wenn das Dokument **nicht** mit einem StyleVision Power Stylesheet verknüpft ist, wird ein Dialogfeld angezeigt, in dem Sie aufgefordert werden, das Dokument mit einem StyleVision Power Stylesheet zu verknüpfen. Falls Sie eine Meldung erhalten, dass keine temporäre Datei (Temp-Datei) erstellt werden konnte, wenn Sie in die Authentic-Ansicht zu wechseln versuchen, wenden Sie sich an Ihren Systemadministrator. Der Systemadministrator muss die Standard-Sicherheits-ID für "non-power User" ändern, damit diesen das Anlegen von Ordnern und Dateien gestattet ist.

Ein- und Ausblenden von XML Markup
In der Authentic-Ansicht können die Tags für alle, einige oder keines der XML-Elemente oder Attribute eingeblendet werden - entweder mit Namen (große Markup-Symbole) oder ohne Namen (kleine Markup-Symbole). Die vier Markup-Symbole werden in der Symbolleiste angezeigt, die entsprechenden Befehle finden Sie im Menü **Authentic.**

Markup-Symbole ausblenden. Alle XML-Tags mit Ausnahme der reduzierten Tags werden ausgeblendet. Wenn Sie im Modus "Markup-Symbole ausblenden" auf einen Tag, der reduziert wurde, doppelklicken (wodurch er normalerweise erweitert wird), wird der Inhalt des Nodes angezeigt und die Tags ausgeblendet.

Kleine Markup-Symbole einblenden. Die XML-Element/Attribut-Tags werden ohne Namen angezeigt.

Große Markup-Symbole einblenden. Die XML-Elemente/Attribute werden mit ihren Namen angezeigt.

Mixed Markup-Symbole einblenden. Im StyleVision Power Stylesheet können Sie für jedes XML-Element oder Attribut einzeln festlegen, wie es im "Mixed Markup"-Modus angezeigt werden soll (mit großen oder kleinen Markup-Symbolen oder gar nicht). Im Mixed Markup-Modus der Authentic-Ansicht ist die Anzeige der Markup-Symbole also benutzerdefiniert. Beachten Sie allerdings, dass diese Einstellungen von der Person vorgenommen werden, die das StyleVision Power Stylesheet erstellt hat.

Bearbeiten von dynamischen Tabellenstrukturen
Zeilen in einer **dynamischen SPS-Tabelle** sind Wiederholungen einer Datenstruktur. Jede

Zeile steht für eine Instanz eines einzelnen Elements, daher hat jede Zeile dieselbe XML-Substruktur wie die darauf folgende.

Die Befehle zum Bearbeiten dynamischer Tabellen dienen zum Bearbeiten der Zeilen einer dynamischen SPS-Tabelle, d.h. Sie können die Anzahl und Reihenfolge der Element-Instanzen ändern. Sie können jedoch nicht die Spalten einer dynamischen SPS-Tabelle bearbeiten, da dadurch die Substruktur der einzelnen Elementinstanzen geändert würde.

Die Symbole zum Bearbeiten dynamischer Tabellen werden in der Symbolleiste angezeigt und können auch über das Menü **Authentic** aufgerufen werden.

Zeile anhängen

Zeile einfügen

Zeile kopieren (d.h. der Inhalt einer Zelle wird dupliziert)

Zeile nach oben

Zeile nach unten

Zeile löschen

Hinweis: Diese Befehle sollten nur bei **dynamischen SPS-Tabellen** und nicht innerhalb von statischen SPS-Tabellen verwendet werden. Die verschiedenen Arten der in der Authentic-Ansicht verwendeten Tabellen werden weiter hinten im Abschnitt Tabellen in der Authentic-Ansicht näher beschrieben.

Erstellen und Bearbeiten von XML-Tabellen
Sie können Ihre eigenen Tabellen einfügen, wenn Sie Daten in Tabellenform darstellen möchten. Solche Tabellen werden als XML-Tabellen eingefügt. Sie können die Struktur einer XML-Tabelle bearbeiten und die Tabelle formatieren. Die Symbole dazu stehen in der Symbolleiste zur Verfügung (siehe Abbildung unten). Im Abschnitt XML-Tabellenbearbeitungsbefehle finden Sie nähere Informationen zu diesen Befehlen.

Die Befehle zu diesen Symbolen **stehen nicht als Menübefehle** zur Verfügung. Beachten Sie außerdem, dass die entsprechende Funktion erst im StyleVision Power Stylesheet aktiviert und konfiguriert werden muss, damit Sie XML-Tabellen verwenden können.

Eine ausführliche Beschreibung der in der Authentic-Ansicht verwendeten Tabellen bzw. wie diese erstellt und bearbeitet werden, finden Sie weiter hinten in der Dokumentation im Abschnitt Tabellen in der Authentic-Ansicht.

Symbole zur Textformatierung
Text wird in der Authentic-Ansicht formatiert, indem man darauf ein XML-Element oder -Attribut anwendet, das die gewünschte Formatierung aufweist. Wurde eine solche Formatierung definiert, kann der Autor des StyleVision Power Stylesheet in der Authentic-Ansicht Symbole zur

Verfügung stellen, mit denen die Formatierung angewendet wird.

Um eine Textformatierung mit Hilfe eines Textformatierungssymbols anzuwenden, markieren Sie den gewünschten Text und klicken Sie auf das entsprechende Symbol.

Das Symbol "Entities definieren"

Mit diesem Symbol wird das Dialogfeld "Entities definieren" geöffnet, in dem der Benutzer der Authentic-Ansicht Entities für die anschließende Verwendung im XML-Dokument definieren kann.

Symbole für die Navigation in DB-Zeilen

Die Pfeilsymbole von links nach rechts sind: Gehe zum ersten Datensatz in der DB; Gehe zum vorhergehenden Datensatz; Dialogfeld "Gehe zu Datensatz #"; Gehe zum nächsten Datensatz; und Gehe zum letzten Datensatz.

Mit diesem Symbol wird das Dialogfeld "Datenbankabfrage bearbeiten" aufgerufen. Hier können Sie eine Abfrage eingeben. In der Authentic-Ansicht werden die abgefragten Datensätze angezeigt.

Das Symbol "XML speichern"

Dient zum Speichern der in der Authentic-Vorschau gemachten Änderungen in einer XML-Arbeitsdatei.

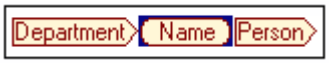
6.3 Authentic-Ansicht Hauptfenster

In der Authentic-Ansicht gibt es vier Ansichtsmodi: Große Markup-Symbole; Kleine Markup-Symbole; Mixed Markup-Symbole; und Alle Markup-Symbole ausblenden. Damit können Sie die Markup-Informationen für Dokumente mit unterschiedlicher Genauigkeit anzeigen.

Um zwischen diesen Ansichtsmodi zu wechseln, verwenden Sie die Befehle im Menü "Authentic" oder die Symbole der Symbolleiste (siehe vorhergehender Abschnitt, Authentic-Ansicht Symbolleistenschaltflächen).

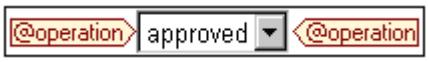
Große Markup-Symbole
In diesem Modus werden die Start- und End-Tags von Elementen und Attributen mit dem Namen des Elements/Attributs im Tag angezeigt:

Das Element Name in der Abbildung oben ist **erweitert**, d.h. sein Start- und End-Tag sowie der Inhalt der Elements werden angezeigt. Durch Doppelklicken auf den Start- oder End-Tag eines Elements/Attributs können Sie dieses **reduzieren.**

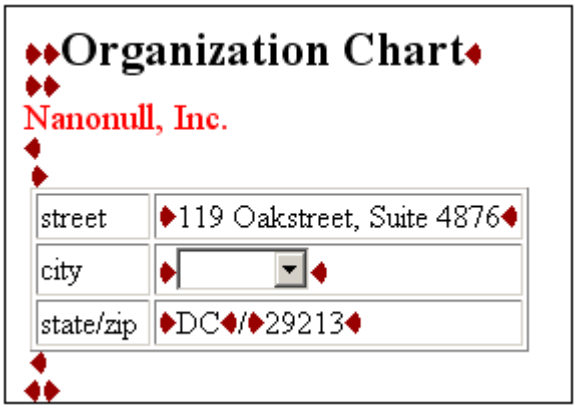


Um das reduzierte Element/Attribut zu erweitern, doppelklicken Sie auf den reduzierten Tag.

Im Modus für große Markup-Symbole werden Attribute durch das Symbol @ im Start- und End-Tag gekennzeichnet:

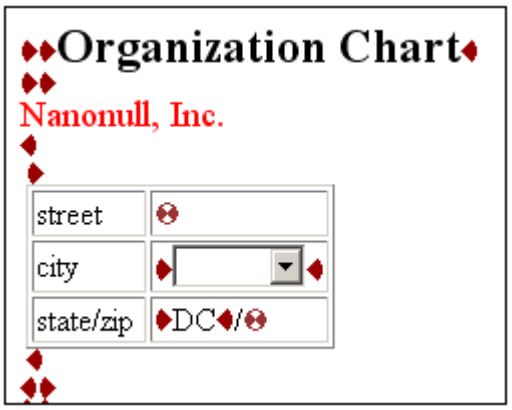


Kleine Markup-Symbole
In diesem Modus werden die Start- und End-Tags von Elementen/Attributen ohne Namen angezeigt:



Um ein Element/Attribut zu reduzieren bzw. zu erweitern, doppelklicken Sie auf den entsprechenden Tag. Im unten gezeigten Beispiel sehen Sie zwei reduzierte Elemente in der Tabelle:

© 2005 Altova GmbH



Mixed Markup-Symbole
Im Mixed Markup-Modus wird eine benutzerdefinierte Markup-Ansicht angezeigt. Der Autor des StyleVision Power Stylesheet kann für einzelne Elemente/Attribute im Dokument große, kleine oder gar keine Markup-Symbole einstellen. Der Benutzer der Authentic-Ansicht sieht diese benutzerdefinierte Ansicht im "Mixed Markup"-Ansichtsmodus.

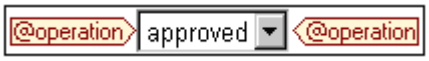
Alle Markup-Symbole ausblenden
In diesem Modus werden alle XML-Markup-Symbole ausgeblendet. Da die Formatierung in der Authentic-Ansicht die Formatierung des gedruckten Dokuments ist, handelt es sich bei diesem Modus um eine WYSIWYG-Ansicht des Dokuments.

Anzeige von Inhalt
In der Authentic-Ansicht wird Inhalt auf zwei Arten angezeigt:

- Reiner Text. Sie geben den Text ein und dieser Text wird der Inhalt des Elements bzw. der Wert des Attributs.

- Dateneingabe-Hilfen. Dabei handelt es sich entweder um ein Eingabefeld (Textfeld), ein Eingabefeld mit mehreren Zeilen, eine Dropdown-Liste, ein Kontrollkästchen oder ein Optionsfeld. Text, den Sie in Eingabefelder und Eingabefelder mit mehreren Zeilen eingeben, wird der XML-Inhalt des Elements bzw. der Wert des Attributs.

  Bei den anderen Dateneingabe-Hilfen wird durch Ihre Auswahl der entsprechende XML-Wert erzeugt, der im StyleVision Power Stylesheet definiert wurde. Wenn also im Beispiel unten "approved" ausgewählt wird, wird dieser Eintrag auf einen XML-Wert "1" oder auf "approved" oder einen beliebigen anderen Eintrag gemappt, während "not approved" dem Wert "0" oder "not approved" oder einem beliebigen anderen Wert zugeordnet wird.



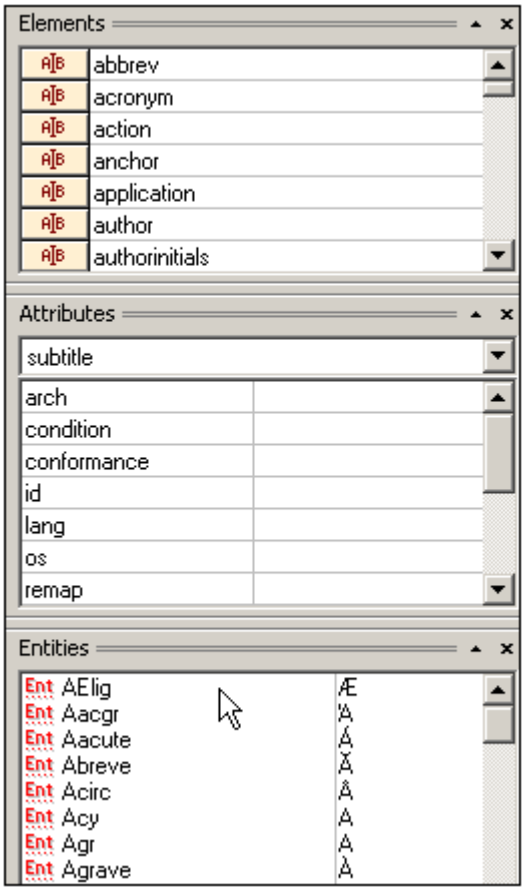
Optionale Nodes
Wenn ein Element oder Attribut gemäß dem referenzierten Schema **optional** ist, wird eine Eingabeaufforderung vom Typ "add [*Element/Attribut*]" angezeigt.

Wenn Sie auf die Eingabeaufforderung klicken, wird das Element hinzugefügt und der Cursor so gesetzt, dass Daten eingegeben werden können. Wenn es mehrere optionale Nodes gibt, wird die Aufforderung "add..." angezeigt. Wenn Sie darauf klicken, wird ein Menü mit den optionalen Nodes angezeigt.

© 2005 Altova GmbH

6.4 Authentic-Ansicht Eingabehilfen

Die Authentic-Ansicht verfügt über drei Eingabehilfen: für Elemente, Attribute und Entities. Sie werden als Fenster am rechten Rand der Authentic-Ansicht angezeigt.



Die Element- und Attribut-Eingabehilfen sind kontextsensitiv, d.h. was in der Eingabehilfe angezeigt wird, hängt davon ab, an welcher Stelle im Dokument der Cursor sich befindet. Die in der Entities-Eingabehilfe angezeigten Entities sind nicht kontextsensitiv; es werden - unabhängig von der Cursorposition - alle im Dokument zulässigen Entities angezeigt.

Im Folgenden werden die einzelnen Eingabehilfen beschrieben.

Element-Eingabehilfe
In der Element-Eingabehilfe werden die Elemente aufgelistet, die hinter dem ausgewählten Element/Textbereich in der Authentic-Ansicht angehängt oder davor eingefügt werden können, innerhalb des Elements/Textbereichs eingefügt oder darauf (d.h. durch Ersetzung) angewendet werden können. Was Sie mit einem in der Eingabehilfe aufgelisteten Element tun können, wird in der Eingabehilfe durch das Symbol auf der linken Seite des Elementnamens angezeigt. Im Folgenden finden Sie eine Liste der in der Element-Eingabehilfe vorkommenden Symbole mit einer Erklärung dazu.

Um ein Element aus der Eingabehilfe zu verwenden, klicken Sie auf sein Symbol.

Hinter Element anhängen
Das Element in der Eingabehilfe wird hinter dem ausgewählten Element angehängt. Beachten Sie, dass es auf der richtigen hierarchischen Ebene

angehängt wird. Wenn sich Ihr Cursor z.B. innerhalb eines `//sect1/para`-Elements befindet und Sie ein `sect1`-Element anhängen, wird das neue `sect1`-Element nicht als Geschwisterelement von `//sect1/para` angehängt, sondern als `sect1`-Element, das ein Parent dieses `para`-Elements ist.

Vor Element einfügen
Das Element in der Eingabehilfe wird vor dem ausgewählten Element eingefügt. Beachten Sie, dass das Element genau wie beim vorhergehenden Befehl auf der richtigen hierarchischen Ebene eingefügt wird.

Element einfügen
Sie können ein Element aus der Eingabehilfe auch **innerhalb** eines Elements einfügen. Wenn Sie den Cursor in ein Element hineinsetzen, können die zulässigen Child-Elemente dieses Elements eingefügt werden. Beachten Sie, dass zulässige Child-Elemente sowohl Teil eines "elements-only Content Model" als auch Teil eines "Mixed Content Model" (Text plus Child-Elemente) sein können.

Ein zulässiges Child-Element kann entweder dann eingefügt werden, wenn ein Textbereich markiert wurde oder wenn der Cursor als Einfügepunkt in den Text platziert wurde.

- Wenn ein Textbereich ausgewählt und ein Element eingefügt wurde, wird der Textbereich der Inhalt des eingefügten Elements.
- Bei Auswahl einer Cursorposition wird das Element an dieser Stelle eingefügt

Nachdem das Element eingefügt wurde, kann es durch Klicken auf eines der beiden Symbole zum Löschen eines Elements (in der Element-Eingabehilfe) gelöscht werden. Welches der beiden Symbole angezeigt wird, ist abhängig davon, ob Sie einen Textbereich auswählen oder ob Sie den Cursor als Einfügepunkt in den Text setzen (siehe unten).

Element anwenden
Wenn Sie in Ihrem Dokument ein Element auswählen (indem Sie in der "Große Markup-Symbol anzeigen"-Ansicht entweder auf den Start- oder den End-Tag eines Elements klicken), bedeutet dieses Symbol, dass das Element in der Eingabehilfe auf das ausgewählte (Original-) Element angewendet werden kann. Das Originalelement wird durch das neu angewendete Element *ersetzt*.

- Wenn das angewendete Element ein **Child-Element mit demselben Namen** wie das Child-Element des Originalelements hat und das Original-Element eine Instanz dieses Child-Elements enthält, dann wird das Child-Element des Originals im neuen Element-Inhalt beibehalten.
- Wenn das angewendete Element **kein Child-Element mit demselben Namen** wie das instantiierte Child-Element des Originalelements enthält, dann wird das instantiierte Child-Element des Originalelements als Geschwisterelement von etwaigen Child-Elementen oder Elementen des neuen Elements angehängt.
- Wenn das angewendete Element **ein Child-Element hat, für das es im Content Model des Originalelements kein Äquivalent gibt**, wird dieses Child-Element nicht direkt erstellt, doch bietet Ihnen die Authentic-Ansicht die Möglichkeit, dieses einzufügen.

Wenn Sie anstatt eines Elements einen Textbereich auswählen, wird bei Anwendung eines Elements auf diesen Bereich an dieser Stelle das

 © 2005 Altova GmbH

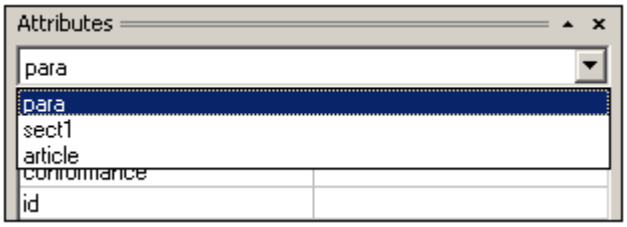
angewendete Element erstellt, wobei der markierte Text dessen Inhalt wird. Wenn der Cursor sich an einem Einfügepunkt befindet, ist die Anwendung eines Elements nicht zulässig.

Element löschen (wenn ein Bereich ausgewählt ist)
Dieses Symbol wird links von dem zeileninternen Element angezeigt, innerhalb dessen sich der ausgewählte Textbereich befindet. Wenn Sie auf dieses Symbol klicken, wird das zeileninterne Element, nicht aber der ausgewählte Textbereich gelöscht.

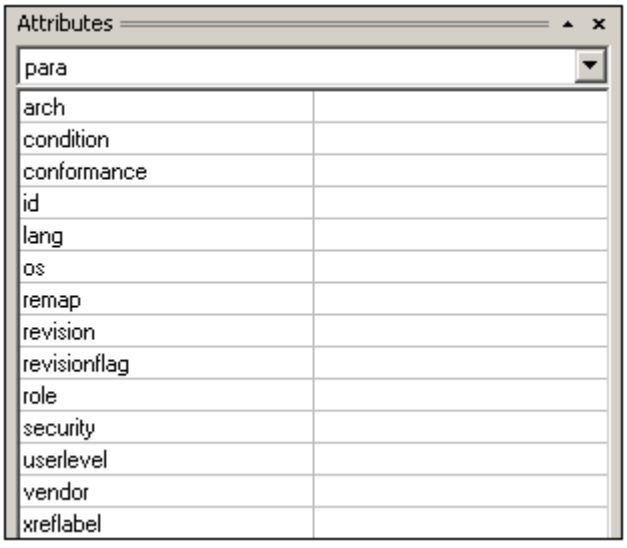
Element löschen (wenn ein Einfügepunkt ausgewählt ist)
Dieses Symbol wird links von dem zeileninternen Element angezeigt, innerhalb dessen sich der Cursor befindet. Wenn Sie auf das Symbol klicken, wird dieses zeileninterne Element gelöscht, nicht aber sein Inhalt.

Attribut-Eingabehilfe
Die Attribut-Eingabehilfe besteht aus einem Listenfeld mit einer Dropdown-Liste von Attributen. Das ausgewählte Element (Sie können auf den Start- oder End-Tag klicken oder den Cursor irgendwo im Element-Inhalt platzieren) wird in der Dropdown-Liste angezeigt.

Die in den nachfolgenden Abbildungen gezeigte Attribut-Eingabehilfe hat ein `para`-Element in der Dropdown-Liste. Wenn Sie auf den Abwärtspfeil in der Liste klicken, erscheint eine Liste aller **übergeordneten Elemente** des `para`-Elements hinauf bis zum Root-Element, in diesem Fall `article`.



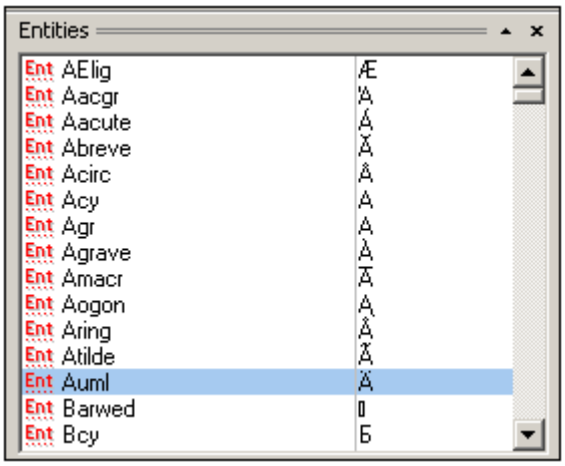
Unterhalb der Dropdown-Liste wird eine Liste gültiger Attribute für dieses Element - in diesem Fall für das `para`-Element angezeigt. Wenn ein Attribut für ein bestimmtes Element obligatorisch ist, wird es fett angezeigt. (Im Beispiel unten gibt es keine obligatorischen Attribute, sondern nur optionale)



Um den Wert für ein Attribut einzugeben, klicken Sie in das Wertefeld des Attributs und geben Sie den Wert ein. Daraufhin wird das Attribut mit dem entsprechenden Wert im XML-Dokument erstellt.
Hinweis: Wenn Sie einen Wert für ein Attribut eingeben, wirkt sich das nur dann auf die Authentic-Ansicht aus, wenn dies im StyleVision Power Stylesheet definiert wurde.

Entities-Eingabehilfe
Über die Entities-Eingabehilfe können Sie eine Entity in Ihr Dokument einfügen. Entities dienen zum Einfügen von Sonderzeichen oder Textfragmenten, die in einem Dokument öfter vorkommen (z.B. der Name einer Firma).

Um eine Entity einzufügen, setzen Sie den Cursor an die gewünschte Stelle im Text und doppelklicken Sie in der Entities-Eingabehilfe auf die Entity.



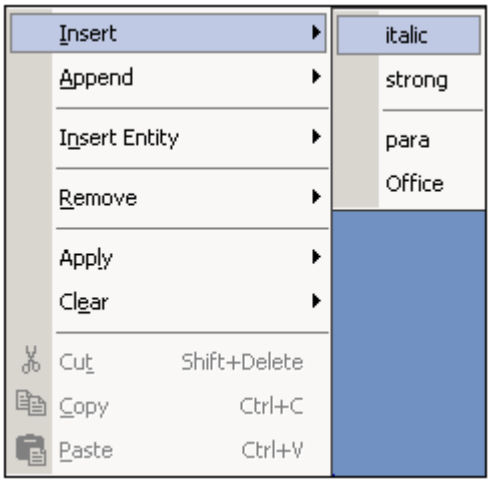
Hinweis: Eine interne Entity ist eine Entitiy, deren Wert in der DTD definiert wurde. Eine externe Entity ist eine Entity, deren Wert in einer externen Quelle z.B. einer anderen XML-Datei enthalten ist. Sowohl interne als auch externe Entities werden in der Entities-Eingabehilfe aufgelistet. Beim Einfügen einer Entity - egal ob intern oder extern - wird die Entity und nicht ihr Wert in den XML-Text eingefügt. Wenn es sich um eine interne Entity handelt, wird in der Authentic-Ansicht **der Wert der Entity** angezeigt. Dies bedeutet, dass eine XML-Datei, die eine externe Entity ist, in der Authentic-Ansicht als Entity angezeigt wird; die Entity wird in der Authentic-Ansicht nicht durch ihren Inhalt ersetzt.

Sie können in der Authentic-Ansicht auch Ihre eigenen Entities definieren: Nähere Informationen dazu finden Sie im Abschnitt "Features und ihre Verwendung" unter Definieren von Entities.

© 2005 Altova GmbH

6.5 Authentic-Ansicht Kontextmenüs

Wenn Sie mit der rechten Maustaste auf einen ausgewählten Dokumenteninhalt oder Node klicken, wird ein Menü mit Befehlen, die für die Auswahl oder Cursorposition relevant sind, angezeigt. Diese Art von Menü wird Kontextmenü genannt. Das angezeigte Kontextmenü hängt davon ab, an welcher Stelle des Dokuments der Cursor sich gerade befindet. In der Abbildung unten sehen Sie das Kontextmenü mit den jeweils aktiven Befehlen. Außerdem wird in der Abbildung das Untermenü "Einfügen" angezeigt, das aus einer Liste von Elementen besteht, die an dieser Stelle eingefügt werden können. Beachten Sie, das das **Untermenü Einfügen** oberhalb der Linie die Elemente anzeigt, die **innerhalb des aktuellen Elements eingefügt werden können** und unterhalb der Linie die Elemente, die **vor dem aktuellen Element eingefügt werden können.** In der Abbildung unten ist das aktuelle Element das `para`-Element. Die Elemente `italic` und `strong` können innerhalb des aktuellen `para`-Elements eingefügt werden. Die Elemente `para` und `Office` können vor dem `para`-Element eingefügt werden.



Die meisten der Befehle aus dem Kontextmenü werden unter Authentic-Ansicht Eingabehilfen beschrieben.

Entfernen eines Elements
Wenn Sie den Cursor über den Befehl **Entfernen** halten, erscheint eine Menüliste bestehend aus dem ausgewählten Element und allen seinen übergeordneten Elementen bis hinauf zum Dokument-Element. Klicken Sie auf das Element, um es zu entfernen. Auf diese Art lässt sich ein Element oder ein beliebiges übergeordnetes Element einfach entfernen. Beachten Sie, dass bei Entfernung eines übergeordneten Elements auch alle untergeordneten Elementen einschließlich des ausgewählten Elements entfernt werden.

6.6 Features und ihre Verwendung

In diesem Abschnitt werden wichtige Features der Authentic-Ansicht näher beschrieben. Bei den Features, die hier behandelt werden, handelt es sich entweder um häufig gebrauchte Funktionen oder solche, deren Funktionsweise und Konzepte näher erklärt werden müssen.

In diesem Abschnitt werden folgende Themen behandelt:

- Es gibt in der Authentic-Ansicht drei verschiedene Tabellenarten. Im Abschnitt Tabellen in der Authentic-Ansicht werden diese drei Tabellenarten (statisches SPS, dynamisches SPS und XML) näher erläutert und deren Verwendung wird erklärt. Die Beschreibung beginnt mit einem allgemeinen Überblick. Anschließend wird näher auf die Einzelheiten eingegangen.
- Die Datumsauswahl ist ein grafischer Kalender, der Daten im korrekten XML-Format eingibt, wenn Sie auf ein Datum klicken. Siehe Datumsauswahl.
- Eine Entity ist ein Kürzel für ein Sonderzeichen oder einen Textstring. Sie können Ihre eigenen Entities definieren, um Sonderzeichen oder Textstrings durch Eingabe der Entity einfügen zu können. Nähere Informationen dazu finden Sie unter Definition von Entities.
- Die Grafikformate, die in der Authentic-Ansicht angezeigt werden können.

6.6.1 Tabellen in der Authentic-Ansicht

Die drei Tabellentypen fallen in zwei Kategorien: SPS-Tabellen (statische und dynamische) und XML-Tabellen.

SPS-Tabellen: Es gibt zwei Arten von SPS-Tabellen: statische und dynamische. SPS-Tabellen werden vom Autor des StyleVision Power Stylesheet, mit dem Ihr XML-Dokument verknüpft ist, erstellt. Sie können selbst keine SPS-Tabelle in das XML-Dokument einfügen, doch können Sie Daten in die Felder der SPS-Tabelle eingeben und Zeilen in dynamischen SPS-Tabellen hinzufügen und löschen. Im nachfolgenden Abschnitt zu SPS-Tabellen werden die Funktionen dieser Tabellen näher beschrieben.

XML-Tabellen werden vom Benutzer der Authentic-Ansicht eingefügt. Auf diese Art können Sie auf Wunsch an jeder Stelle im Dokument, an der dies zulässig ist, Tabellen einfügen. Die Editierfeatures von XML-Tabellen und die Bearbeitungssymbole für XML-Tabellen werden im Folgenden beschrieben.

SPS-Tabellen

In der Authentic-Ansicht werden zwei Arten von SPS-Tabellen verwendet: statische Tabellen und dynamische Tabellen.

Statische Tabellen sind von der Struktur und der Art des Inhalts der Zellen unveränderbar. Als Benutzer der Authentic-Ansicht können Sie Daten in die Tabellenzellen eingeben, die Struktur der Tabellen (d.h. Hinzufügen von Zeilen oder Spalten usw.) oder den Inhaltstyp einer Zelle jedoch nicht verändern. Die Eingabe von Daten erfolgt entweder durch Eingabe von Text oder durch Auswahl von Optionen in Form von Kontrollkästchen, Optionsfeldern oder Dropdown-Listen. Nachdem Sie Daten eingegeben haben, können Sie diese bearbeiten.

© 2005 Altova GmbH

Nanonull, Inc.

Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

Hinweis: Die Symbole oder Befehle zum Bearbeiten dynamischer Tabellen **dürfen nicht** zum Bearbeiten statischer Tabellen verwendet werden.

Dynamische Tabellen enthalten Zeilen, die eine sich wiederholende Datenstruktur darstellen, d.h. jede Zeile weist dieselbe Datenstruktur auf (Dies ist bei statischen Tabellen nicht der Fall). Sie können daher Zeilen bearbeiten, also Zeilen anhängen, einfügen, nach oben oder nach unten verschieben und löschen. Diese Befehle stehen im Menü **Authentic** und als Symbole in der Symbolleiste (siehe Abb. unten) zur Verfügung.

Um diese Befehle zu verwenden, platzieren Sie den Cursor an eine beliebige Stelle in der gewünschten Zeile und wählen Sie den entsprechenden Befehl aus.

Administration

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

Zum Navigieren innerhalb der Zellen einer Tabelle verwenden Sie die Pfeiltasten der Tastatur (nach oben, nach unten, nach links, nach rechts). Um von einer Zelle zur nächsten zu springen, drücken Sie die Tabulatortaste. Wenn Sie in der letzten Zelle einer Zeile die Tabulatortaste drücken, wird eine neue Zeile erstellt.

XML-Tabellen

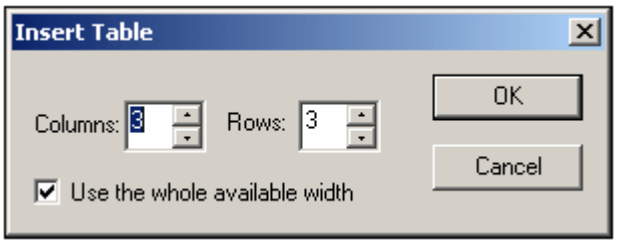
XML-Tabellen können von Ihnen, dem Benutzer der Authentic-Ansicht verwendet werden. Sie können damit Tabellen überall im XML-Dokument, wo dies zulässig ist, einfügen. Diese Funktion eignet sich zum Einfügen von Tabellendaten in Ihr Dokument. Diese Tabellen werden als Tabellen gedruckt, wenn Sie den Druckauftrag direkt von der Authentic-Ansicht aus geben. Wenn Sie Ausgabedokumente auch mit Hilfe von XSLT Stylesheets erzeugen, besprechen Sie

die gewünschte Ausgabe mit dem Autor des StyleVision Power Stylesheet.

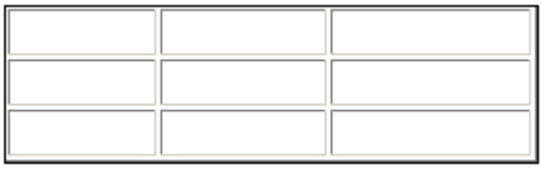
Beachten Sie bitte, dass XML-Tabellen nur an bestimmten Stellen, an denen dies zulässig ist, eingefügt werden können. Dies ist im Schema (DTD oder XML Schema) definiert. Wenn Sie Tabellen an anderen Stellen einfügen möchten, muss dies mit dem für das StyleVision Power Stylesheet Verantwortlichen besprochen werden.

Arbeiten mit XML-Tabellen
Die Arbeit mit XML-Tabellen erfordert drei Schritte: Einfügen der Tabelle, Formatierung der Tabelle und Eingabe von Daten. Die Befehle für die Arbeit mit XML-Tabellen stehen in Form von Symbolen in der Symbolleiste zur Verfügung (siehe Symbole zur Bearbeitung von XML-Tabellen).

Einfügen von Tabellen
Um eine XML-Tabelle einzufügen, setzen Sie Ihren Cursor an die Stelle, an der die Tabelle eingefügt werden soll und klicken Sie auf das Symbol . (Beachten Sie dass XML-Tabellen nur an bestimmten im Schema definierten Stellen eingefügt werden können). Das Dialogfeld "Tabelle einfügen" (siehe unten) wird nun angezeigt.



Wählen Sie die Anzahl der Spalten und Zeilen aus und geben Sie an, ob die Tabelle sich über die gesamte Breite erstrecken soll. Daraufhin wird anhand der Angaben oben die folgende Tabelle erstellt.

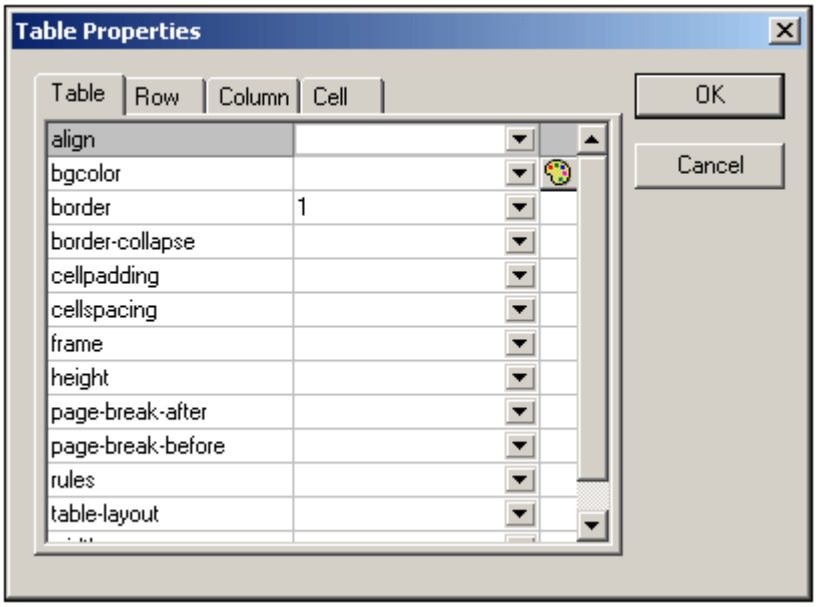


Sie können später Spalten hinzufügen und löschen, Zeilen erstellen und Spalten zusammenfügen. Erstellen Sie daher zuerst die grundlegende Struktur.

Hinweis: Alle Änderungen an der Tabellenstruktur müssen mit Hilfe der Befehle des Tabellenmenüs vorgenommen werden. Sie können nicht durch Änderung von Attributwerten in der Attribut-Eingabehilfe gemacht werden.

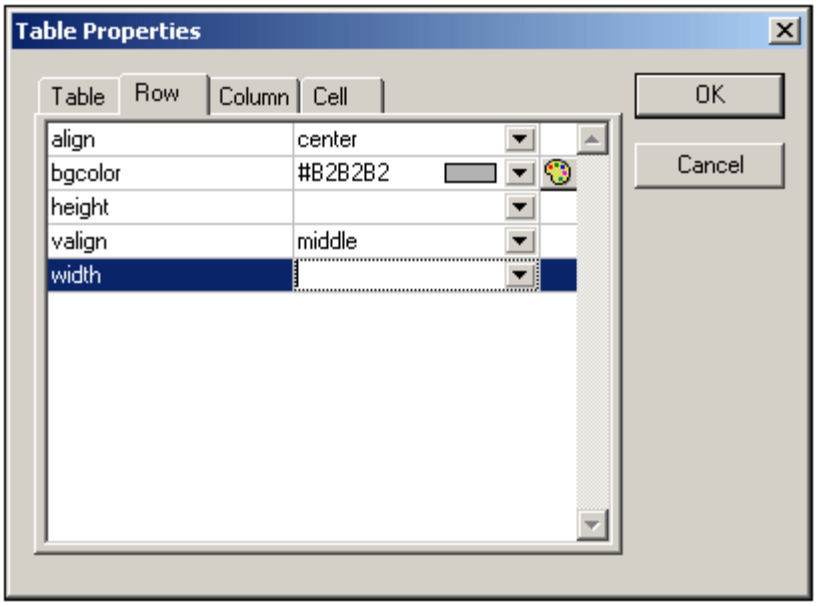
Formatieren von Tabellen und Eingabe von Daten
Platzieren Sie Ihren Cursor zum Formatieren der Tabelle an eine beliebige Stelle in der Tabelle und klicken Sie auf das Symbol (Tabelleneigenschaften). Daraufhin erscheint das Dialogfeld "Tabelleneigenschaften" (siehe unten), in dem Sie die Formatierung für die Tabelle, oder für einzelne Zeilen, Spalten oder Zellen definieren können.

 © 2005 Altova GmbH



Setzen Sie die Eigenschaften "cellspacing" und "cellpadding" auf "0". Ihre Tabelle wird nun folgendermaßen aussehen:

Setzen Sie nun den Cursor in die erste Zeile, um diese zu formatieren und klicken Sie auf das Symbol (Tabelleneigenschaften). Klicken Sie auf das Register "Row".



Da es sich bei der ersten Zeile um die Kopfzeile handelt, definieren Sie eine Hintergrundfarbe, um diese Zeile von den anderen Zeilen zu unterscheiden. Beachten Sie dass die Zeileneigenschaften in der Abbildung oben definiert wurden. Geben Sie anschließend den Text

für die Spaltenüberschriften ein. Ihre Tabelle sieht nun folgendermaßen aus:

Name	Telephone	Email

Beachten Sie, dass die Ausrichtung wie angegeben mittig (centered) ist. Angenommen, Sie möchten nun die Spalte "Telephone" in die Spalten "Office" und "Home" unterteilen. Um dies zu tun, müssen Sie Zellen teilen. Setzen Sie die Cursor in die Zelle "Telephone" und klicken Sie auf das Symbol (Vertikal teilen). Ihre Tabelle sieht nun aus wie folgt:

Name	Telephone	Email

Setzen Sie nun den Cursor in die Zelle unterhalb der Zelle mit dem Eintrag "Telephone" und klicken Sie auf das Symbol (Horizontal teilen). Geben Sie dann die Spaltenüberschriften "Office" und "Home" ein. Ihre Tabelle sieht nun folgendermaßen aus:

Name	Telephone		Email
	Office	Home	

Nun müssen Sie die einzelnen Zellen in der Spalte "Telephone" vertikal teilen.

Mit Hilfe der Symbole zum Bearbeiten von Tabellen können Sie des Weiteren Spalten und Zeilen hinzufügen und löschen und Zellinhalte vertikal ausrichten. Die Symbole zum Bearbeiten von XML-Tabellen sind in der Benutzerreferenz im Abschnitt "XML-Tabellensymbole" näher beschrieben.

Navigieren innerhalb von Tabellenzellen
Um innerhalb der Zellen der XML-Tabelle navigieren zu können, verwenden Sie die Pfeiltasten der Tastatur.

Eingabe von Daten in eine Zelle
Um Daten in eine Zelle einzugeben, setzen Sie den Cursor in eine Zelle und geben Sie die Daten ein.

Formatieren von Text
Text in einer XML-Tabelle muss wie anderer Text im XML-Dokument mit Hilfe von XML-Elementen oder Attributen formatiert werden. Um ein Element hinzuzufügen, markieren Sie den Text und doppelklicken Sie auf das gewünschte Element in der Element-Eingabehilfe. Um einen Attributwert zu definieren, setzen Sie den Cursor in das Textfragment und geben Sie den erforderlichen Attributwert in der Attribut-Eingabehilfe ein. Nachdem Sie den Kopfzeilentext fett formatiert haben, sieht Ihre Tabelle folgendermaßen aus:

 © 2005 Altova GmbH

Name	Telephone		Email
	Office	Home	

Der Text in der Abbildung oben wurde durch Anklicken und Doppelklick auf das Element `strong` formatiert, für das eine globale Vorlage vorhanden ist, in der als Schriftbreite "Fett" definiert ist. Die Textformatierung wird sofort sichtbar.

Hinweis: Damit die Textformatierung in der Authentic-Ansicht angezeigt wird, muss in StyleVision für das entsprechende Element eine globale Vorlage mit der erforderlichen Textformatierung erstellt worden sein.

Symbole zum Bearbeiten von XML-Tabellen

Die Befehle zum Bearbeiten von XML-Tabellen stehen in Form von Symbolen in der Symbolleiste zur Verfügung. Nachfolgend finden Sie eine Liste dieser Befehle. Beachten Sie, dass es zu diesen Symbolen keine Menübefehlentsprechungen gibt.

Ausführliche Informationen zur Verwendung von XML-Tabellen finden Sie unter XML-Tabellen.

Tabelle einfügen

Der Befehl "Tabelle einfügen" fügt eine **CALS / HTML-Tabelle** an der aktuellen Cursorposition ein.

Tabelle löschen

Der Befehl "Tabelle löschen" löscht die derzeit aktive Tabelle.

Zeile anhängen

Der Befehl "Zeile anhängen" hängt eine Zeile an das Ende der derzeit aktiven Tabelle an.

Spalte anhängen

Der Befehl "Spalte anhängen" hängt eine Spalte an das Ende der derzeit aktiven Tabelle an.

Zeile einfügen

Der Befehl "Zeile einfügen" fügt eine Zeile oberhalb der aktuellen Cursorposition in der derzeit aktiven Tabelle ein.

Spalte einfügen

Der Befehl "Spalte einfügen" fügt eine Spalte links von der derzeit aktiven Cursorposition in der derzeit aktiven Tabelle ein.

Links zusammenfügen

Der Befehl "Links zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle links davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Rechts zusammenfügen

Der Befehl "Rechts zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle rechts davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Unten zusammenfügen

Der Befehl "Unten zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle unterhalb davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Oben zusammenfügen

Der Befehl "Oben zusammenfügen" fügt die aktive Zelle (an der Cursorposition) mit der Zelle oberhalb davon zusammen. Die Tags von beiden Zellen bleiben in der neuen Zelle erhalten, die Spaltenüberschriften bleiben unverändert.

Horizontal teilen

Der Befehl "Horizontal teilen" erstellt eine neue Zelle rechts von der derzeit aktiven Zelle. Die Größe der beiden Zellen entspricht nun der Größe der Originalzelle

Vertikal teilen

Der Befehl "Vertikal teilen" erstellt eine neue Zelle unterhalb von der derzeit aktiven Zelle.

Bündig oben

Dieser Befehl richtet den Inhalt der Zellen bündig am oberen Zellenrand aus.

Vertikal zentrieren

Dieser Befehl zentriert den Inhalt der Zelle.

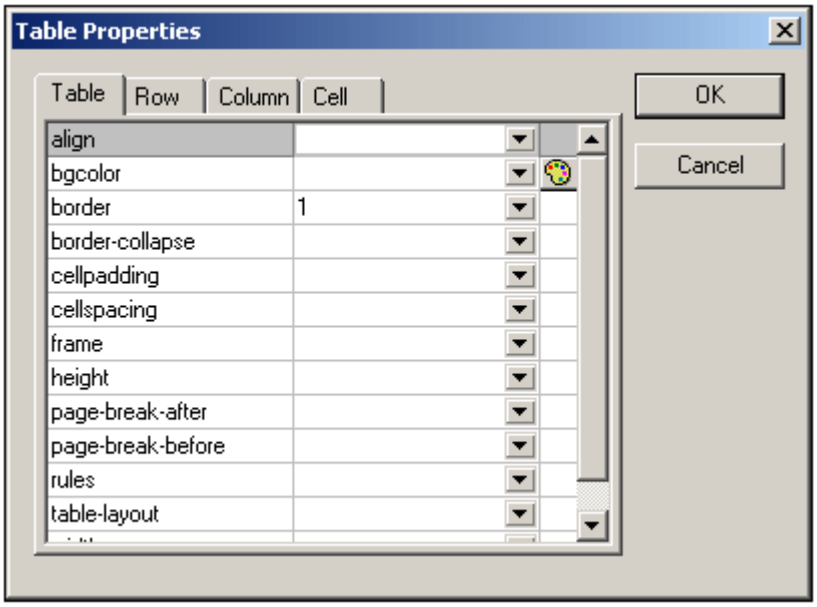
Bündig unten

Dieser Befehl richtet den Inhalt der Zellen bündig am unteren Zellenrand aus.

Tabelleneigenschaften

Der Befehl "Tabelleneigenschaften" öffnet das Dialogfeld "Tabelleneigenschaften". Dieses Symbol wird nur bei HTML-Tabellen aktiv, bei CALS-Tabellen ist es deaktiviert.

© 2005 Altova GmbH



6.6.2 Bearbeiten einer DB

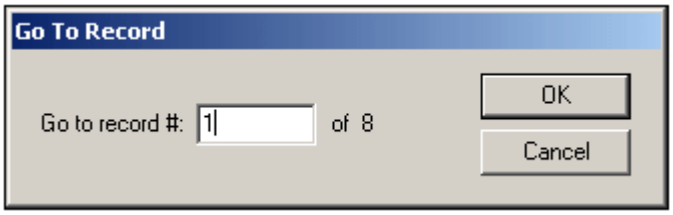
In der Authentic-Ansicht können Sie Datenbank (DB)-Tabellen bearbeiten und die Daten in einer DB speichern. Dieser Abschnitt enthält eine ausführliche Beschreibung der Funktionen, die Ihnen zum Bearbeiten einer DB-Tabelle zur Verfügung stehen. Beachten Sie bitte die folgenden Punkte:

- Die Anzahl der Datensätze in einer DB-Tabelle, die in der Authentic-Ansicht angezeigt werden können, wurde unter Umständen vom Autor des StyleVision Power Stylesheet absichtlich eingeschränkt, um das Design kompakter zu gestalten. In diesem Fall wird anfangs nur diese beschränkte Anzahl an Datensätzen in der Authentic-Ansicht geladen. Sie können die restlichen Datensätze der DB-Tabelle mit Hilfe der Symbole zur Navigation in DB-Tabellenzeilen (*siehe* Navigieren in einer DB-Tabelle) laden und anzeigen.
- Sie können eine Datenbankabfrage durchführen, um bestimmte Datensätze anzuzeigen.
- Sie können Datensätze hinzufügen, ändern und löschen und Ihre Änderungen in der DB speichern. Siehe Ändern einer DB-Tabelle.

Navigieren in einer DB-Tabelle

Die Befehle zum Navigieren in den Zeilen einer DB-Tabelle stehen in Form von Schaltflächen im Authentic-Ansichtsdokument zur Verfügung. Normalerweise gibt es zu jeder DB-Tabelle einen Navigationsbereich mit vier oder fünf Schaltflächen.

Die Pfeilsymbole von links nach rechts sind: Gehe zum ersten Datensatz in der DB; Gehe zum vorhergehenden Datensatz; Dialogfeld "Gehe zu Datensatz #" öffnen (*siehe Abbildung unten*); Gehe zum nächsten Datensatz; und Gehe zum letzten Datensatz.



Um durch eine DB-Tabelle zu navigieren, klicken Sie auf die gewünschte Schaltfläche.

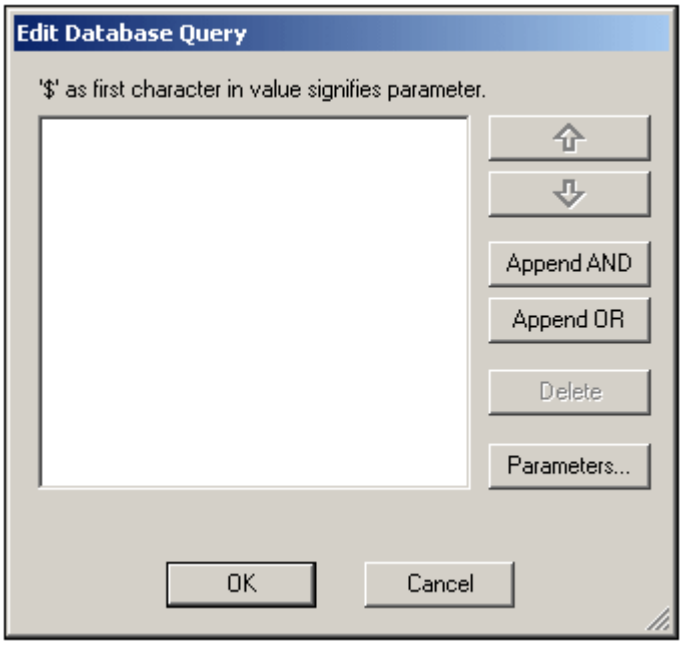
DB-Abfragen

Mit Hilfe einer DB-Abfrage können Sie die Datensätze einer in der Authentic-Ansicht angezeigten Tabelle abfragen. Eine Abfrage wird für eine einzelne Tabelle erstellt. Es kann für jede Tabelle nur eine Abfrage erstellt werden. Sie können während der Bearbeitung eines Dokuments jederzeit eine Abfrage erstellen. Wenn das in der Authentic-Ansicht angezeigte Dokument zu dem Zeitpunkt, zu dem Sie die Abfrage absenden, nicht gespeicherte Änderungen enthält, werden Sie gefragt, ob **alle** Änderungen im Dokument gespeichert oder verworfen werden sollen. Beachten Sie, dass auch an anderen Tabellen vorgenommene Änderungen gespeichert/verworfen werden. Nach Absenden der Abfrage wird die Tabelle mit den aktuellen Abfragekriterien erneut geladen.

Hinweis: Falls eine Meldung angezeigt wird, dass zu viele Tabellen geöffnet sind, können Sie die Anzahl der offenen Tabellen mit Hilfe einer Abfrage durch Filtern reduzieren.

So erstellen Sie eine Abfrage und senden diese ab:

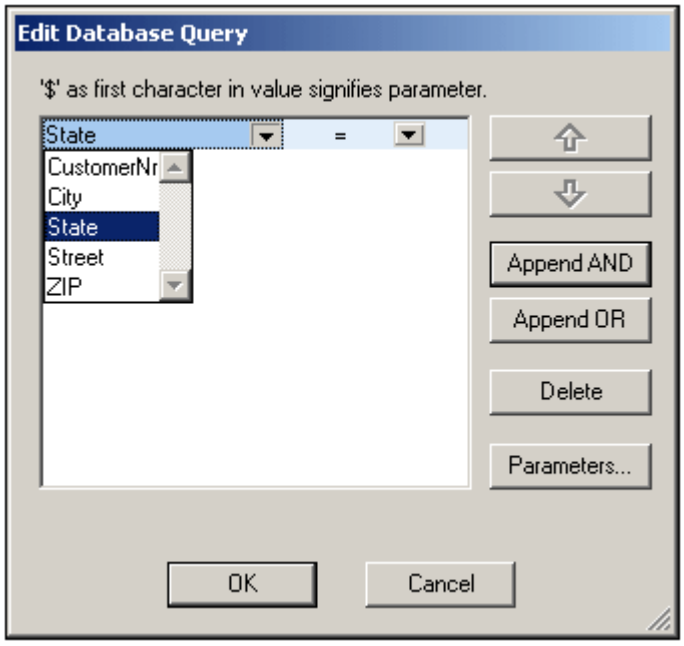
1. Klicken Sie auf die Schaltfläche "Abfrage" für die entsprechende Tabelle, um das Dialogfeld "Datenbankabfrage bearbeiten" aufzurufen (*siehe Abbildung unten).* Diese Schaltfläche erscheint normalerweise am oberen Rand einer DB-Tabelle oder unterhalb davon. Wenn keine Abfrage-Schaltfläche vorhanden ist, hat der Autor des StyleVision Power Stylesheet die DB-Abfragefunktion für diese Tabelle nicht aktiviert.



2. Klicken Sie auf die Schaltfläche **AND anhängen** oder **OR anhängen**. Daraufhin wird

 © 2005 Altova GmbH

ein leeres Abfragekriterium (siehe unten) angehängt.



4. Geben Sie den Ausdruck für das Kriterium ein. Ein Ausdruck besteht aus: (i) einem Feldnamen (der in der dazugehörigen Dropdown-Liste zur Verfügung steht); (ii) einem Operator (der in der zugehörigen Dropdown-Liste zur Verfügung steht); und (iii) einem Wert (direkt einzugeben). Nähere Informationen zum Erstellen von Ausdrücken finden Sie im Abschnitt Ausdrücke in Kriterien weiter unten.
5. Wenn Sie ein weiteres Kriterium hinzufügen möchten, klicken Sie - je nachdem, welchen logischen Operator (AND oder OR) Sie verwenden wollen, um Ihre Kriterien zu verknüpfen - auf die Schaltfläche **AND anhängen** bzw. **OR anhängen.** Fügen Sie nun das neue Kriterium hinzu. Nähere Informationen zu logischen Operatoren finden Sie im Abschnitt Neuordnen von Kriterien in DB-Abfragen.

Ausdrücke in Kriterien
Ausdrücke in DB-Abfragekriterien bestehen aus einem Feldnamen, einem Operator und einem Wert. Als **Feldnamen** stehen die Child-Elemente der obersten Ebene der Datentabelle zur Verfügung; die Namen dieser Felder werden in der Dropdown-Liste aufgelistet (*siehe Abbildung oben)*. Unterhalb sehen Sie eine Liste der verfügbaren **Operatoren**:

=	Ist gleich
<>	Ist nicht gleich
<	Kleiner als
<=	Kleiner oder gleich
>	Größer als
>=	Größer oder gleich
LIKE	Phonetisch ähnlich
NOT LIKE	Phonetisch nicht ähnlich
IS NULL	Ist leer
NOT NULL	Ist nicht leer

Wenn IS NULL oder NOT NULL ausgewählt ist, ist das Wertefeld deaktiviert. **Werte** müssen ohne Anführungszeichen (oder andere Begrenzungszeichen) eingegeben werden. Außerdem müssen Werte dieselbe Formatierung wie die des entsprechenden DB-Felds haben, sonst ist das Ergebnis des Ausdrucks FALSE. Wenn z.B. ein Kriterium für ein Feld des `date`-Datentyps in einer MS Access DB einen Ausdruck `StartDate=25/05/2004` enthält, ist das Ergebnis der Abfrage FALSE, da der `date`-Datentyp in einer MS Access DB das Format `JJJJ-MM-TT` hat.

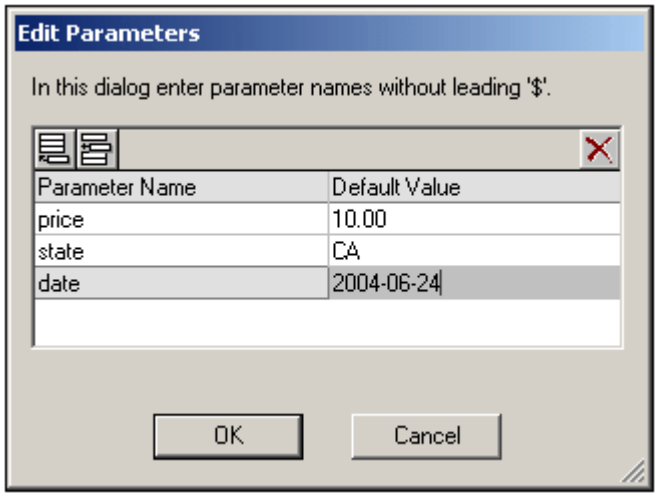
Verwendung von Parametern mit DB-Abfragen
Sie können auch den Namen eines Parameters als Wert eines Ausdrucks eingeben. In diesem Fall wird der Parameter aufgerufen und sein Wert wird als der Wert dieses Ausdrucks verwendet. Dabei kann es sich um einen Parameter handeln, der bereits für das Stylesheet deklariert wurde oder um einen Parameter, den Sie in der Folge für die Verwendung in einem Ausdruck deklarieren.

Parameter sind nützlich, wenn Sie einen einzelnen Wert in mehreren Ausdrücken verwenden wollen.

Um den Namen eines Parameters als Wert eines Ausdrucks zu verwenden, geben Sie das $-Zeichen in das Werteingabefeld ein, gefolgt vom Namen des Parameters (ohne Leerzeichen). Wenn der Parameter bereits deklariert wurde, wird der Eintrag grün angezeigt. Wenn der Parameter noch nicht deklariert wurde, erscheint der Eintrag rot und der Parameter muss deklariert werden.

Deklarieren von Parametern über das Dialogfeld "Datenbankabfrage bearbeiten"
Um das Dialogfeld "Parameter bearbeiten" (zum Deklarieren von Parametern) aufzurufen, gehen Sie folgendermaßen vor:

1. Klicken Sie im Dialogfeld "Datenbankabfrage bearbeiten" auf die Schaltfläche **Parameter...**, um das unten gezeigte Dialogfeld **Parameter bearbeiten** aufzurufen.



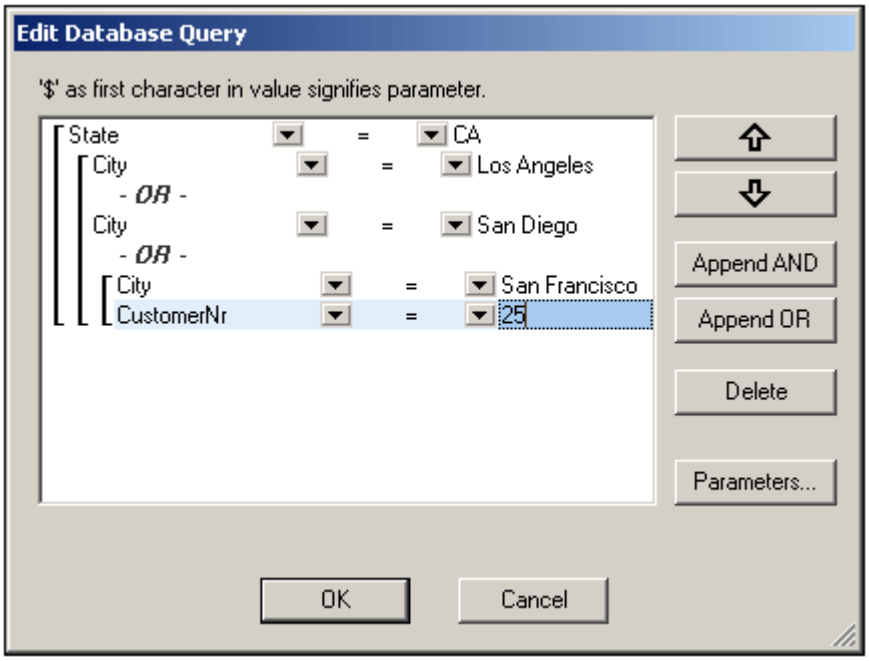
2. Geben Sie den Namen und Wert des Parameters in die entsprechenden Felder ein.

Hinweis: Das Dialogfeld "Parameter bearbeiten" enthält **alle** Parameter, die für das Stylesheet definiert wurden. Es ist zwar ein Fehler, im StyleVision Power Stylesheet einen nicht deklarierten Parameter zu verwenden, doch ist es kein Fehler, einen Parameter zu deklarieren und diesen nicht zu verwenden.

Neuordnen von Kriterien in DB-Abfragen
- Die logische Struktur der DB-Abfrage und die Beziehung zwischen zwei Kriterien oder Kriteriengruppen wird grafisch angezeigt. Jede Ebene der logischen Struktur wird durch eine

 © 2005 Altova GmbH

eckige Klammer angezeigt. Zwei Kriterien oder Kriteriengruppen nebeneinander bedeuten, dass der AND-Operator verwendet wurde, während bei Trennung von zwei Kriterien durch das Wort OR der OR-Operator verwendet wurde. Die Kriterien dienen auch dazu, dem Benutzer einen klaren Überblick über die logische Struktur der DB-Abfrage zu verschaffen.



Die in der obigen Abbildung gezeigte DB-Abfrage würde in Textform folgendermaßen aussehen:
`State=CA AND (City=Los Angeles OR City=San Diego OR (City=San Francisco AND CustomerNr=25))`

Sie können die DB-Abfrage umstrukturieren, indem Sie ein Kriterium oder eine Gruppe von Kriterien im Vergleich zu den anderen Kriterien in der DB-Abfrage nach oben oder unten verschieben. Um ein Kriterium oder eine Gruppe von Kriterien zu verschieben, gehen Sie vor wie folgt:

- Wählen Sie das Kriterium durch Anklicken aus oder wählen Sie eine ganze Ebene aus, indem Sie auf die Klammer für diese Ebene klicken.
- Klicken Sie im Dialogfeld auf den Nach oben- oder Nach unten-Pfeil.

Beachten Sie die folgenden Punkte:

- Wenn sich das in Richtung der Verschiebung benachbarte Kriterium auf derselben Ebene befindet, werden die beiden Kriterien vertauscht.
- Eine Gruppe von Kriterien (d.h. ein Kriterium mit einer Klammer) ändert seine Position innerhalb derselben Ebene; die Ebene ändert sich nicht.
- Ein einzelnes Kriterium ändert seine Position innerhalb derselben Ebene. Wenn sich das benachbarte Kriterium weiter außen/innen (d.h. nicht auf derselben Ebene) befindet, wird das ausgewählte Kriterium **immer um eine Ebene** nach außen/innen verschoben.

Um ein Kriterium in einer DB-Abfrage zu löschen, wählen Sie es aus und klicken Sie auf **Löschen**.

Ändern einer DB-Abfrage

Um eine DB-Abfrage zu ändern, klicken Sie auf die Schaltfläche "Abfrage" , um das Dialogfeld "Datenbankabfrage bearbeiten" aufzurufen. Sie können nun die Ausdrücke in jedem

der aufgelisteten Kriterien ändern, neue Kriterien hinzufügen oder Kriterien aus der DB-Abfrage löschen. Nachdem Sie Ihre Änderungen vorgenommen haben, klicken Sie auf OK. Die Daten aus der DB werden automatisch in StyleVision neu geladen, um die Änderungen an der DB-Abfrage anzuzeigen.

Ändern einer DB-Tabelle

Hinzufügen eines Datensatzes

Um einen Datensatz zu einer DB-Tabelle hinzuzufügen, setzen Sie den Cursor in die gewünschte Zeile der DB-Tabelle und klicken Sie auf das Symbol , (um eine Zeile anzuhängen) oder (um eine Zeile einzufügen). Daraufhin wird ein neuer Datensatz in der temporären XML-Datei erstellt. (Der neue Datensatz wird erst dann in der DB erstellt, wenn Sie den Befehl **Datei | Speichern...** auswählen) In der Authentic-Ansicht wird in der Anzeige der DB-Tabelle eine neue Zeile für den neuen Datensatz angehängt. Der `AltovaRowStatus` für diesen Datensatz wird auf `A` (für Added = Hinzugefügt) gesetzt.

Wenn Sie Daten für den neuen Datensatz eingeben, werden diese fett und unterstrichen angezeigt. Auf diese Art können Sie neu hinzugefügte Datensätze von vorhandenen unterscheiden - wenn vorhandene Datensätze nicht mit diesen Texteigenschaften formatiert wurden. Datentypfehler werden rot gekennzeichnet.

Der neue Datensatz wird zur DB hinzugefügt, wenn Sie **Datei | Speichern...** auswählen. Nachdem ein neuer Datensatz zur Datenbank hinzugefügt wurde, wird das `AltovaRowStatus`-Feld initialisiert (gekennzeichnet durch `---`) und der Datensatz wird in der Authentic-Ansicht als regulärer Datensatz angezeigt.

Ändern eines Datensatzes

Um einen Datensatz zu ändern, setzen Sie den Cursor an die gewünschte Stelle in der DB-Tabelle und nehmen Sie die gewünschten Änderungen vor. Wenn die Anzahl der Datensätze beschränkt ist, müssen Sie eventuell (mit Hilfe der oben beschriebenen Navigationssymbole) zum gewünschten Datensatz navigieren.

Wenn Sie einen Datensatz bearbeiten, werden die Einträge in allen Feldern des Datensatzes unterstrichen und der `AltovaRowStatus` aller primären Instanzen dieses Datensatzes wird auf `U` (für Updated = Aktualisiert) gesetzt. Für alle sekundären Instanzen dieses Datensatzes wird der `AltovaRowStatus` auf `u` (klein geschrieben) gesetzt. Die primären und sekundären Instanzen eines Datensatzes werden durch die Struktur der DB definiert - und ebenfalls von dem davon generierten XML-Schema. Wenn z.B. eine Adresstabelle in einer Kundentabelle enthalten ist, kann die Adresstabelle im Design-Dokument in zwei Instantiierungstypen vorkommen: als Adresstabelle selbst und innerhalb von Instantiierungen der Kundentabelle. Derjenige der beiden Typen, der geändert wird, ist der primär geänderte Typ. Andere Typen - es kann mehr als einen anderen Typ geben - sind sekundäre Typen. Datentypfehler werden rot markiert.

Sie speichern die Änderungen in der DB durch Klicken auf **Datei | Speichern...**. Nachdem ein geänderter Datensatz in der DB gespeichert wurde, wird das `AltovaRowStatus` Feld initialisiert (gekennzeichnet durch `---`) und der Datensatz wird in der Authentic-Ansicht als regulärer Datensatz angezeigt.

Hinweis:

- Selbst wenn nur ein einziges Feld eines Datensatzes in der Authentic-Ansicht geändert wird, wird der gesamte Datensatz beim Speichern der Daten in der Datenbank aktualisiert.

 © 2005 Altova GmbH

- Der Datumswert `0001-01-01` ist bei einigen Datenbanken als `NULL`-Wert definiert und könnte daher eine Fehlermeldung verursachen.

Löschen eines Datensatzes
Um einen Datensatz zu löschen, setzen Sie den Cursor in die Zeile für den entsprechenden Datensatz und klicken Sie auf das Symbol . Der zu löschende Datensatz erscheint nun durchgestrichen. Der `AltovaRowStatus` erhält folgenden Wert: Primäre Instanzen des Datensatzes werden auf `D gesetzt;` sekundäre Instanzen auf `d`; indirekt gelöschte Datensätze werden mit `X` gekennzeichnet. Indirekt gelöschte Datensätze sind Felder im gelöschten Datensatz, die in einer separaten Tabelle enthalten sind. So könnte z.B. in einer Kundentabelle eine Adresstabelle enthalten sein. Wenn ein Kundendatensatz gelöscht würde, würde indirekt auch der entsprechende Adressdatensatz gelöscht. Wenn in der Kundentabelle ein Adressdatensatz gelöscht würde, würde der Adressdatensatz in der Kundentabelle primär gelöscht, jedoch würde derselbe Datensatz in einer unabhängigen Adresstabelle sekundär gelöscht, wenn diese instantiiert würde.

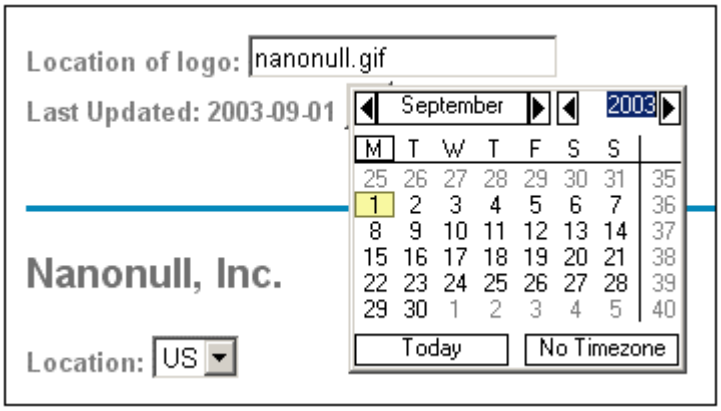
Sie speichern die Änderungen in der DB durch Klicken auf **Datei | Speichern...**.

Hinweis: Wenn Sie Daten in der DB speichern, wird der Rückgängig-Befehl zurückgesetzt. Sie können Operationen, die vor dem Speichern durchgeführt wurden, nicht mehr rückgängig machen.

6.6.3 Datumswahl

Die Datumswahl ist ein grafischer Kalender zur Eingabe eines Datums in einem Standardformat in ein XML-Dokument. Es ist für die Verarbeitung des Dokuments wichtig, dass ein Standardformat verwendet wird. Das Datumswahlsymbol erscheint neben dem Datumsfeld, das dadurch geändert wird (siehe Abbildung unten).

Um die Datumswahl (siehe unten) anzuzeigen, klicken Sie auf das Datumswahl-Symbol.



Um ein Datum auszuwählen, klicken Sie auf das gewünschte Datum, den gewünschten Monat bzw. das gewünschte Jahr. Das Datum wird in das XML-Dokument eingegeben und das angezeigte Datum wird entsprechend geändert. Falls nötig können Sie auch eine Zeitzone angeben.

6.6.4 Definieren von Entities

Unabhängig davon, ob Ihr Dokument auf einer DTD oder einem XML-Schema basiert, können Sie für die Verwendung in der Authentic-Ansicht Entities definieren. Diese Entities werden anschließend in der Entities-Eingabehilfe und im Kontextmenü-Untermenü von **Entity einfügen** angezeigt. Wenn Sie in der Entities-Eingabehilfe auf eine Entity doppelklicken, wird diese an der Cursorposition eingefügt.

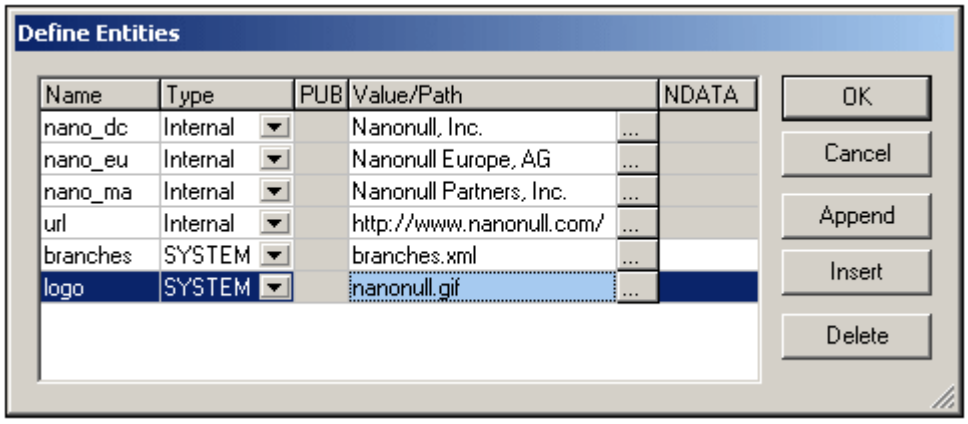
Eine Entity eignet sich v.a. dann, wenn Sie einen Textstring, ein XML-Fragment oder einige andere externe Ressourcen an mehreren Stellen in Ihrem Dokument verwenden möchten. Die Entity, bei der es sich im Grunde um eine Kurzbezeichnung für die benötigten Daten handelt, wird im Dialogfeld "Entities definieren" definiert. Nachdem Sie die Entity definiert haben, können Sie diese an mehreren Stellen in Ihrem Dokument verwenden und dadurch Zeit sparen und effizienter arbeiten.

Sie können in Ihrem Dokument zwei grundlegende Arten von Entities verwenden: **geparste Entities**, wobei es sich um XML-Daten handelt (entweder einen Textstring oder ein Fragment aus einem XML-Dokument) oder **ungeparste Entities**, wobei es sich um nicht-XML-Daten handelt wie z.B. eine Binärdatei (normalerweise eine Grafik, Ton oder ein Multimedia-Objekt). Jede Entity hat einen Namen und einen Wert. Im Fall von geparsten Entities ist die Entity ein Platzhalter für die XML-Daten. Der Wert der Entity sind entweder die XML-Daten selbst oder eine URI, die auf eine `.xml`-Datei verweist, die die XML-Daten enthält. Bei ungeparsten Entities ist der Wert der Entity eine URI, die auf eine nicht-XML-Datei verweist.

So definieren Sie eine Entity:

1. Klicken Sie auf **Authentic | XML-Entities definieren...**. Daraufhin wird das Dialogfeld "Entities definieren" geöffnet.

© 2005 Altova GmbH



2. Geben Sie den Namen Ihrer Entity im Feld "Name" ein. Dies ist der Name, der in der Entities-Eingabehilfe angezeigt werden wird.
3. Wählen Sie den Typ der Entity aus der Dropdown-Liste im Feld "Type" aus. Es stehen drei Typen zur Auswahl. Eine **interne** (internal) Entity ist eine Entity, für die der zu verwendende Text im XML-Dokument selbst gespeichert ist. Wenn Sie **PUBLIC** oder **SYSTEM** auswählen, wird angegeben, dass sich die Ressource außerhalb der XML-Datei befindet und mittels eines "Public Identifier" oder eines "System Identifier" gefunden wird. Ein System Identifier ist eine URI, die den Pfad der Ressource definiert. Ein Public Identifier ist ein pfadunabhängiger Identifier, der es manchen Prozessoren ermöglicht, die Ressource zu identifizieren. Wenn Sie sowohl einen Public als auch einen System Identifier angeben, wird der Public Identifier zum System Identifier aufgelöst und es wird der System Identifier verwendet.
4. Wenn Sie als Type PUBLIC ausgewählt haben, geben Sie den Public Identifier der Ressource im Feld PUBLIC ein. Wenn Sie Internal oder SYSTEM als Type ausgewählt haben, ist das Feld PUBLIC deaktiviert.
5. Im Feld "Value/Path" haben Sie folgende Eingabemöglichkeiten:
   - Wenn die Entity "Internal" ist, geben Sie den Textstring ein, der als Wert der Entity verwendet werden soll. Setzen Sie den Textstring nicht in Anführungszeichen. Alle Anführungszeichen werden als Teil des Textstrings behandelt. Beachten Sie, dass Entities eine gute Methode zur Eingabe von Unicode-Zeichen in Ihr Dokument sind; geben Sie dazu die Unicode-Nummer als Wert einer internen Entity ein.
   - Wenn der Entity-Typ SYSTEM ist, geben Sie die URI der Ressource ein oder wählen Sie mit Hilfe der Durchsuchen-Schaltfläche eine Ressource auf Ihrem lokalen Netzwerk aus. Wenn die Ressource geparste Daten enthält, muss es sich um eine XML-Datei handeln (d.h. die Datei muss die Erweiterung `.xml` haben). Als Alternative dazu kann die Ressource eine Binärdatei sein, z.B. eine GIF-Datei.
   - Wenn der Entity-Typ PUBLIC ist, müssen Sie zusätzlich einen System Identifier in dieses Feld eingeben.
6. Der Eintrag NDATA sagt dem Prozessor, dass diese Entity nicht geparst, sondern an den entsprechenden Prozessor gesendet werden soll. Das Feld "NDATA" sollte daher nur mit ungeparsten Entities verwendet werden.

Dialogfeldfunktionen
Durch Klicken auf die entsprechende Schaltfläche können Sie Entities anhängen, einfügen und löschen. Außerdem können Sie Entities spaltenweise alphabetisch sortieren, indem Sie auf die entsprechende Spaltenüberschrift klicken; bei einem einmaligen Klick wird aufsteigend sortiert, bei zweimaligem Klick absteigend. Sie können außerdem die Größe des Dialogfelds und die Spaltenbreite anpassen.

Einschränkungen

- Eine Entity innerhalb einer anderen Entity wird nicht aufgelöst - weder im Dialogfeld, noch in der Authentic-Ansicht, noch in der XSLT-Ausgabe, und das kaufmännische Und-Zeichen einer derartigen Entity wird in seiner Escape-Form, also `&` angezeigt.
- Externe Entities werden in der Authentic-Ansicht nicht aufgelöst, es sei denn es handelt sich bei der Entity um eine Bilddatei und sie wurde als der Wert eines Attributs vom Typ `ENTITY` oder `ENTITIES` eingegeben. Derartige Entities werden aufgelöst, wenn das Dokument mit einem anhand des StyleVision Power Stylesheet generierten XSLT verarbeitet wird.

6.6.5 Grafiken in der Authentic-Ansicht

Die Authentic-Ansicht basiert auf dem Internet Explorer und kann die meisten der Grafikformate, die von Ihrer Internet Explorer-Version unterstützt werden, anzeigen. Unterstützt werden die folgenden gebräuchlichen Grafikformate:

- GIF
- JPG
- PNG
- BMP
- WMF (Microsoft Windows Metafile)
- EMF (Enhanced Metafile)
- SVG (nur für PDF-Ausgabe)

6.6.6 Verwendung von Tasten in der Authentic-Ansicht

Eingabetaste
In der Authentic-Ansicht wird die **Eingabetaste** an bestimmten Cursorpositionen verwendet, um zusätzliche Elemente anzuhängen. Wenn das Kapitel eines Buchs z.B. (laut Schemadefinition) mehrere Absätze enthalten kann, wird bei Drücken der **Eingabetaste** innerhalb des Texts des Absatzes ein neuer Absatz direkt hinter dem aktuellen Absatz angehängt. Wenn ein Kapitel einen Titel und mehrere Kapitel enthalten kann, wird bei Drücken der **Eingabetaste** innerhalb des Kapitels aber außerhalb eines Absatzelements (auch innerhalb des Titel-Elements) ein neues Kapitel hinter dem aktuellen Kapitel angehängt (vorausgesetzt es sind laut Schemadefinition mehrere Kapitel zulässig).

Hinweis: Bei Drücken der **Eingabetaste** wird keine Zeilenschaltung, kein Zeilenvorschub eingefügt, d.h. es wird keine neue Zeile angelegt. Dies ist auch dann der Fall, wenn sich der Cursor innerhalb eines Textnode wie z.B. eines Absatzes befindet.

© 2005 Altova GmbH

7 Browser-Ansicht

Die Browser-Ansicht wird normalerweise zur Ansicht folgender Dateien verwendet:

- XML-Dateien, die mit einer XSLT-Datei verknüpft wurden. Wenn Sie in die Browser-Ansicht wechseln, wird die XML-Datei mit Hilfe eines XSLT Stylesheet on-the-fly transformiert. Das Ergebnis wird direkt in der Browser-Ansicht angezeigt.
- HTML-Dateien, die direkt im HTML-Format erstellt wurden oder mittels einer XSLT-Transformation einer XML-Datei erzeugt wurden.

Für die Browser-Ansicht wird Microsoft Internet Explorer 5.0 oder höher benötigt. Zur Anzeige von XML-Dateien, die mit Hilfe eines XSLT Stylesheet transformiert wurden, sollten Sie unbedingt Internet Explorer 6.0 oder höher verwenden, da diese Versionen MSXML 3.0 verwenden, einen XML-Parser, die den XSLT 1.0 Standard vollständig unterstützt. Eventuell sollten Sie auch MSXML 4.0 installieren. Nähere Informationen dazu erhalten Sie in unserem Download Center. (Beachten Sie, dass die Unterstützung von XSL in IE 5 nicht zu 100 % kompatibel mit der offiziellen XSLT Recommendation ist. Wenn Sie daher in der Browser-Ansicht Probleme mit IE 5 haben, sollten Sie auf IE 6 upgraden.)

Um XML-und HTML-Dateien in der Browser-Ansicht zu sehen, klicken Sie auf das Register "Browser".

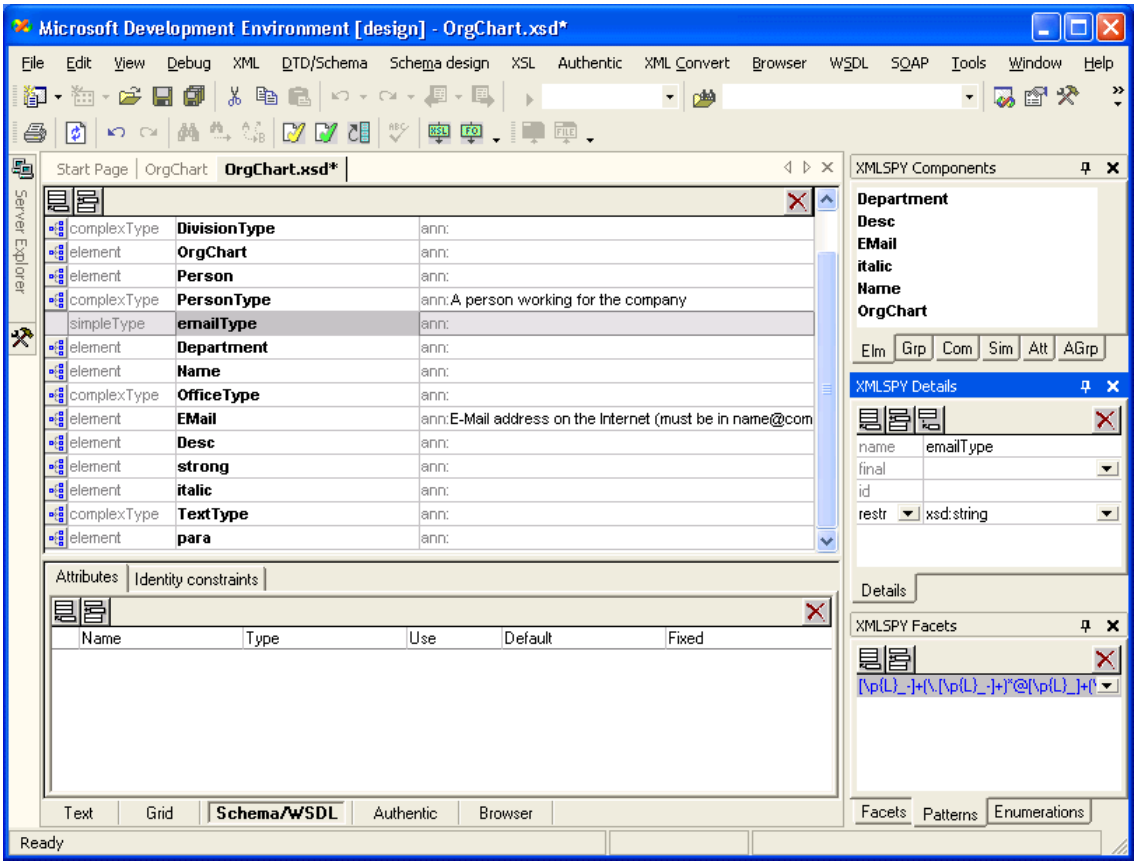
Funktionen der Browser-Ansicht

- Die Browser-Ansicht lässt sich in einem separaten Fenster öffnen. Wechseln Sie dazu in die Browser-Ansicht und wählen Sie den Menübefehl **Browser | Neues Fenster**. Auf diese Art können Sie Fenster nebeneinander anordnen und so die Browser-Ansicht Seite an Seite mit der Bearbeitungsansicht anzeigen. Alle Änderungen, die Sie in der Bearbeitungsansicht vornehmen, sind sofort in der Browser-Ansicht sichtbar: Drücken Sie in der Bearbeitungsansicht einfach die Taste **F5** oder machen Sie die Browser-Ansicht zum aktiven Fenster (indem Sie darauf klicken).
- Die Browser-Ansicht unterstützt die Suchfunktion. Wählen sie dazu in der Browser-Ansicht die Menübefehl **Bearbeiten | Suchen**, um Textstrings zu suchen.
- Die Browser-Ansicht unterstützt gebräuchliche Browserbefehle: Zurück, Vorwärts, Abbrechen, Aktualisieren, Schriftgröße und Drucken.

8 XMLSpy 2005 in MS Visual Studio .NET

Sie können Ihre XMLSpy 2005 Version in die Microsoft Visual Studio .NET IDE-Versionen 2002 und 2003 integrieren. Auf diese Weise stehen Ihnen sowohl die hoch entwickelten XML-Editierfunktionen als auch die professionelle Entwicklungsumgebung von Visual Studio .NET zur Verfügung:

- Installieren Sie Microsoft Visual Studio .NET
- Installieren Sie XMLSpy 2005 (Enterprise oder Professional Edition)
- Laden Sie das XMLSpy 2005 Integrationspaket für Microsoft Visual Studio .NET herunter und starten Sie es. Sie finden das Installationspaket auf der XMLSpy 2005 (Enterprise und Professional Edition) Download-Seite unter www.altova.com. (**Hinweis:** Sie müssen das Integrationspaket für Ihre jeweilige Version von XMLSpy 2005 (Enterprise oder Professional) verwenden).



Nach Installation des Integrationspakets können Sie XMLSpy 2005 innerhalb der Visual Studio .NET-Umgebung verwenden.

© 2005 Altova GmbH

8.1 Unterschiede zwischen der .NET und der Standalone-Version

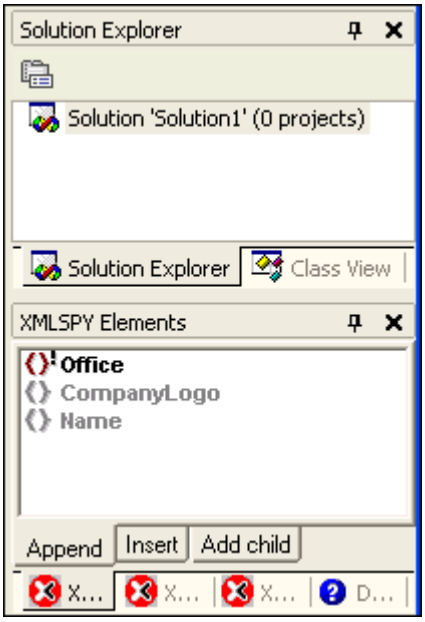
Die Enterprise und die Professional XMLSpy 2005 Visual Studio .NET Edition verfügen in Visual Studio .Net 2002 und 2003 über dieselben Funktionalitäten.

XMLSpy 2005-Menüs oder -Menüoptionen, die beim Start von Visual Studio .NET angezeigt werden:

- File | New | AUTHENTIC
- XML-Konvertierung
- SOAP
- Tools | XMLSpy Optionen

Eingabehilfen (Toolfenster in Visual Studio .NET)

- Die Eingabehilfen-Fenster werden beim Start unterhalb des Projektmappen-Explorers auf einer Registerkarte gruppiert.
- Sie können die Eingabehilfen-Fenster an jede beliebige Stelle in der Entwicklungsumgebung ziehen.
- Wenn Sie auf ein Eingabehilfen-Register rechtsklicken, können Sie die Schnittstelle weiter anpassen. Sie können die Eingabehilfen-Fenster als abgedockt, angedockt, ausgeblendet und automatisch ausgeblendet konfigurieren.



Geänderte Funktionen in den Visual Studio .NET Editions:

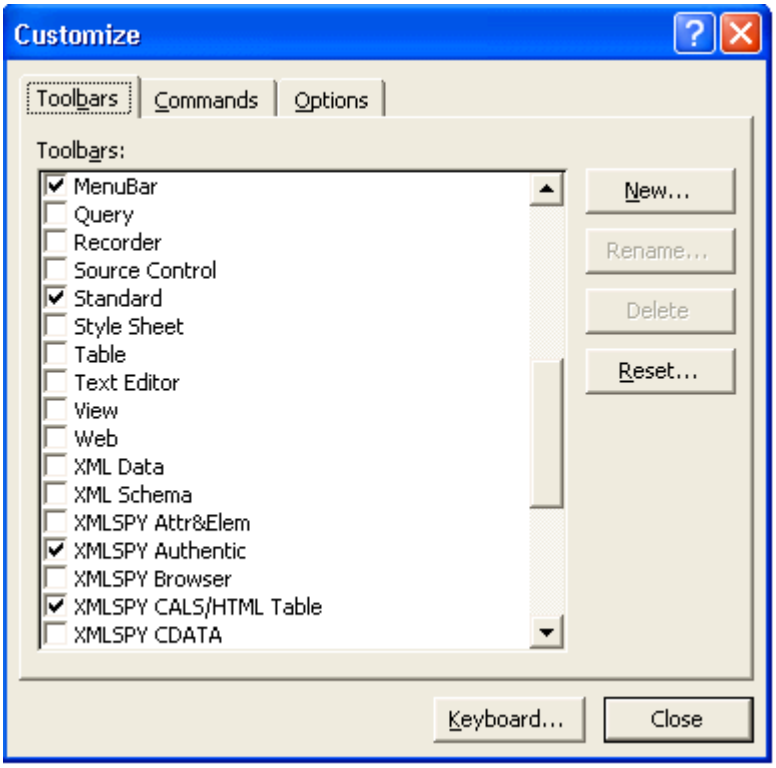
File-Menü
Verwenden Sie die Befehle **File | Open | File from web** anstelle von **File | URL öffnen** um eine Datei über eine URL zu öffnen.
Verwenden Sie die Befehle **File | Save "my file" as | My network places** anstelle von **File | Speichern unter URL** um eine Datei unter einer URL zu speichern.

Edit-Menü **,** Undo und Redo
Die Befehle Undo und Redo wirken sich auf alle Aktionen (Kopieren, Einfügen usw.) aus, die in der Entwicklungsumgebung und in XMLSpy 2005 durchgeführt wurden.

Bitte beachten Sie, dass es zwei Symbolsätze zum Rückgängigmachen und Wiederherstellen einer Aktion gibt. Die XMLSpy 2005-Symbole befinden sich in der XMLSpy-Hauptsymbolleiste.

Die Visual Studio .NET-Symbole sind deaktiviert, solange Dateien in XMLSpy 2005 geöffnet sind.

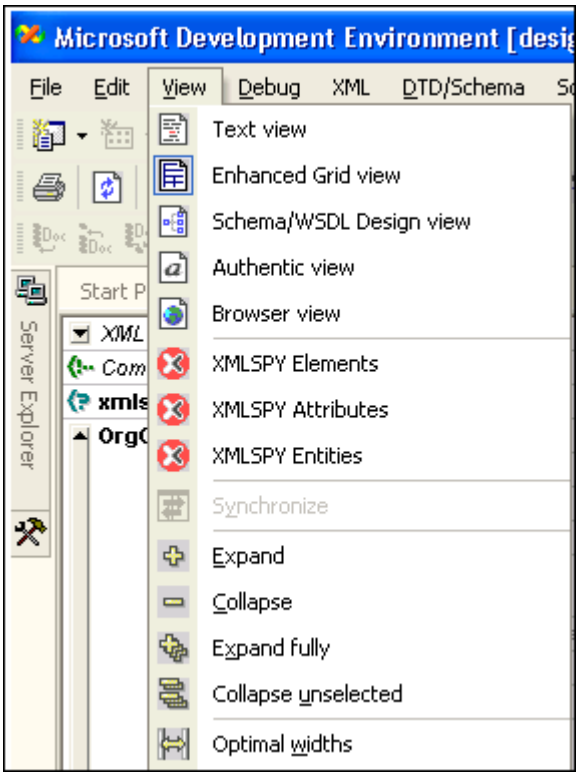
Menü **Tools | Customize | Toolbar**, **Commands** und **Options**.



Diese Registerkarten enthalten sowohl Visual Studio .NET- als auch XMLSpy 2005-Befehle.

View-Menü

© 2005 Altova GmbH



Im Menü View (Ansicht) können Sie Folgendes auswählen bzw. aufrufen:

- die Art der Ansicht, in der Ihr XML-Dokument angezeigt werden soll: Text-, Enhanced Grid-, Schema/WSDL-, Authentic- oder Browser-Ansicht.
- eventuell zuvor geschlossene XMLSpy 2005-Eingabehilfen-Fenster.
- XSLT- und SOAP Debugger Tool-Fenster, die unter Umständen während eines Debugger-Vorgangs geschlossen wurden.
- Funktionen der Grid-Ansicht zur übersichtlicheren Anzeige von Daten.

Help-Menü
Das Hilfemenü enthält das Untermenü XMLSpy 2005 Hilfe, über das Sie die XMLSpy 2005-Hilfe aufrufen können. Des Weiteren finden Sie darin Links zum Altova Support Center, einen Bereich zum Downloaden von Komponenten usw.

Nicht unterstützte Features

Info-Fenster
Das Info-Fenster wird nicht unterstützt. Dieses Fenster enthält zusätzliche Informationen zu ausgewählten Elementen/Attributen usw., d.h. es enthält den Namen des Elements/Attributs sowie Informationen zu datatype, enumeration und occurrence.

Projekt-Fenster und **Projekt-**Menü (sowie die Versionskontroll-Untermenüs).
Das Projekt-Menü (und die Versionskontroll-Untermenüs) stehen nicht zur Verfügung, da MS Visual Studio .NET seine eigene Projekt- und Versionskontroll-Umgebung hat. Aus diesem Grund werden die Validierung und Umwandlung von Dateien im Batch-Modus derzeit nicht unterstützt.

Extras | Skripting-Umgebung...
Die Skripting-Umgebung steht derzeit nicht zur Verfügung.

Eigenes Browser-Fenster
Das Kontrollkästchen "In eigenem Fenster anzeigen" im Menü **Extras | Optionen |** Register

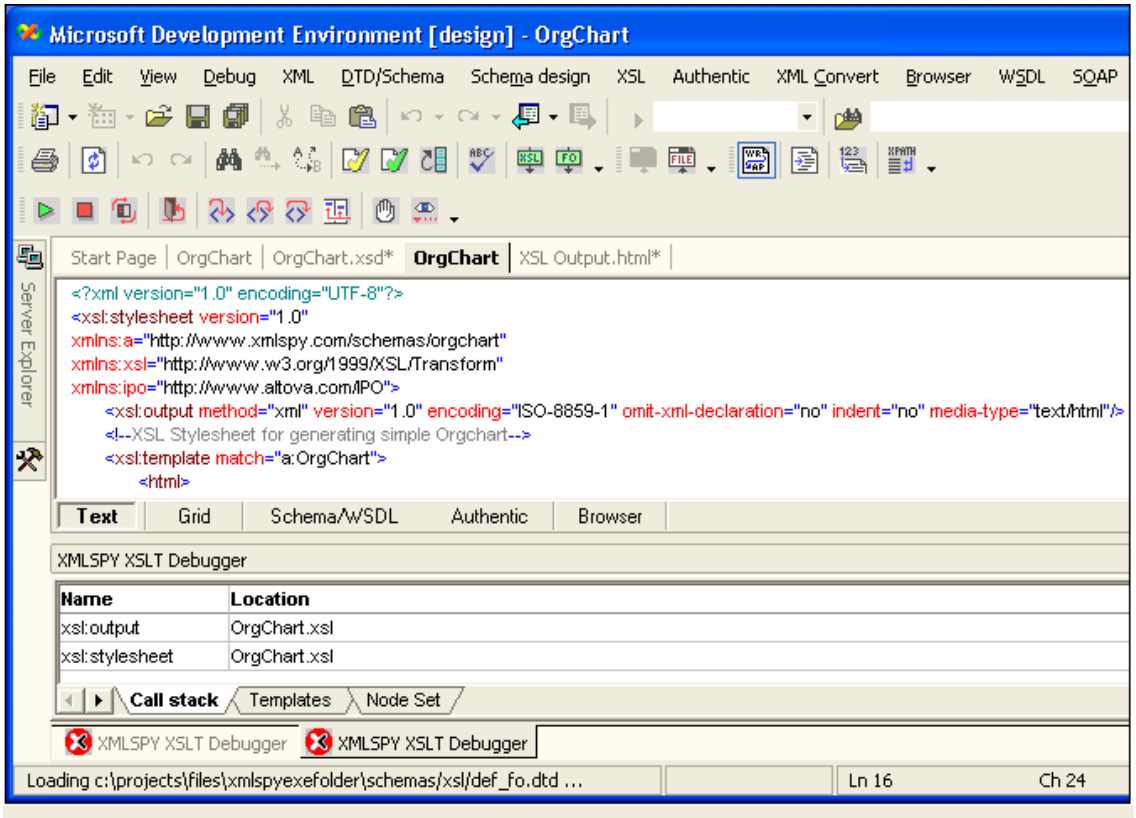
Ansicht wird nicht unterstützt, daher werden die Text- und die Browser-Ansicht immer in ein und demselben Fenster angezeigt - z.B. beim Transformieren einer XML-Datei in eine HTML-Datei.

Authentic-Ansicht
Symbole zum Textstatus sind in der Authentic-Ansicht nicht verfügbar. Die Funktionen bzw. die Formatierung, die Sie damit aktivieren, können jedoch weiterhin über die Eingabehilfe oder das Kontextmenü aufgerufen werden. Nähere Informationen dazu finden Sie in der Dokumentation zu Stylesheet Designer.

© 2005 Altova GmbH

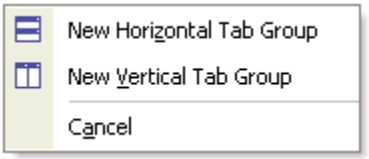
8.2 Visual Studio .NET und XSLT Debugger

Im unten gezeigten Screenshot sehen Sie, wie die Datei `OrgChart.xml` angezeigt wird, wenn der XSLT Debugger aktiviert ist. Standardmäßig werden alle Dateien auf einer Registerkartengruppe angezeigt. Um den Debugging-Vorgang übersichtlicher zu gestalten, können Sie in Visual Studio .NET eine eigene Registerkarte anlegen.

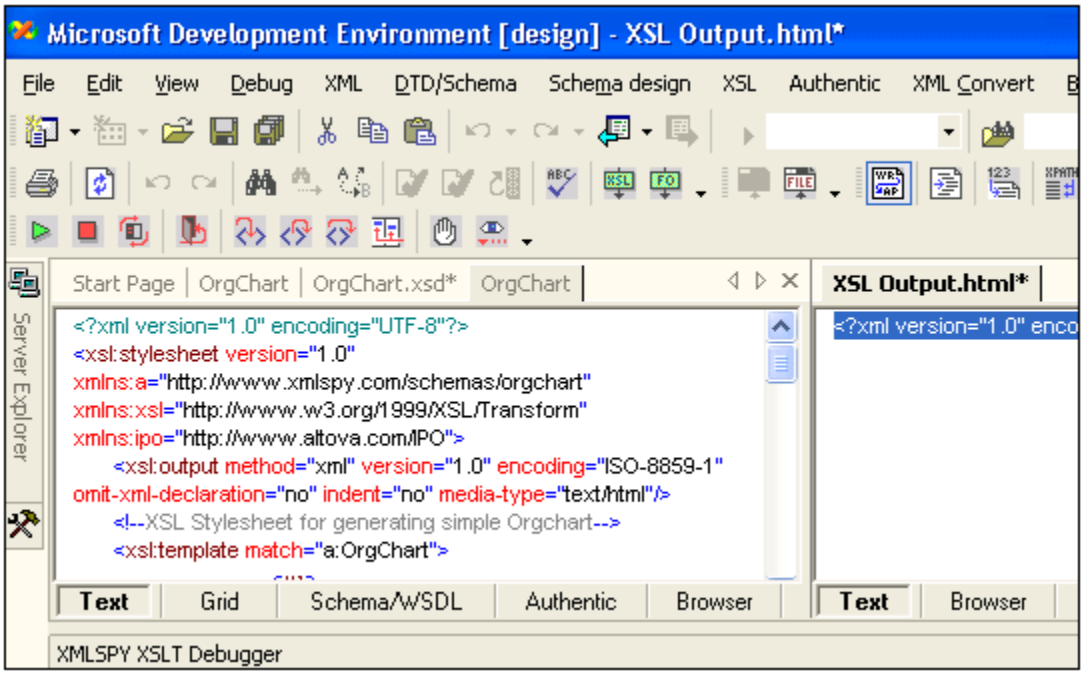


So legen Sie eine neue Registerkartengruppe an:

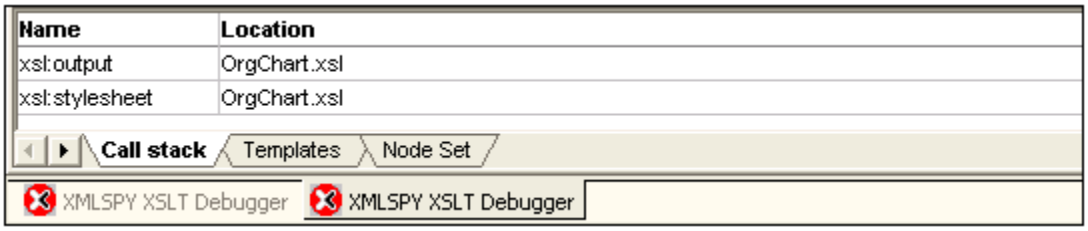
1. Klicken Sie auf das Register **XSL Output.html** und ziehen Sie es auf die aktive Registerkarte.
2. Hierauf wird ein Popup-Menü angezeigt, in dem Sie die Art der zu erstellenden Registerkarte definieren können.



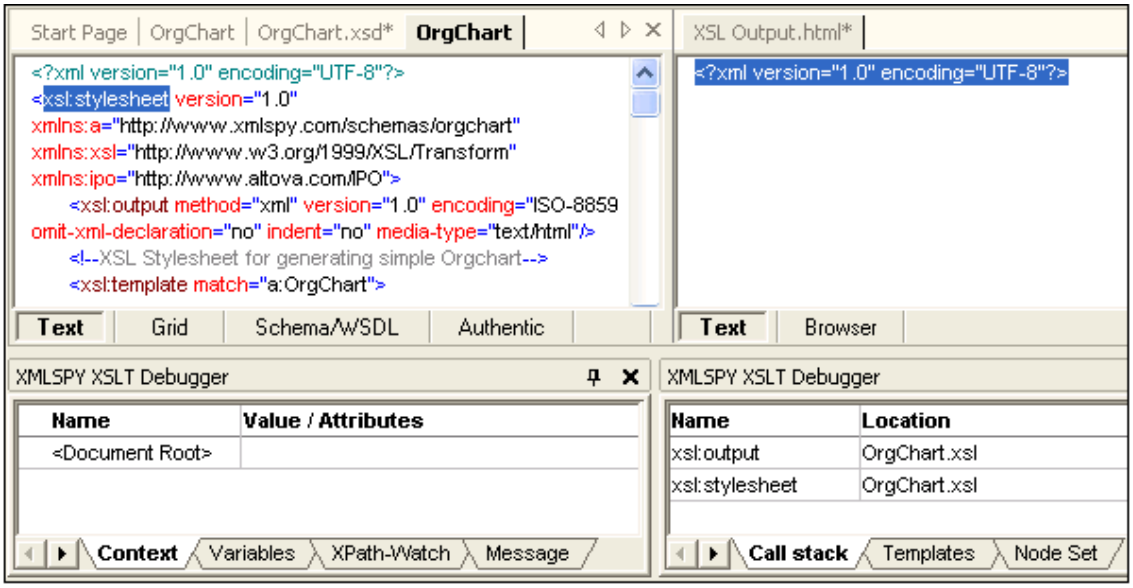
3. Wählen Sie die Option "New Vertical Tab Group".
   Daraufhin wird eine neue Registerkarte mit der Datei XSL Output.html erstellt.



4. Erstellen Sie mit derselben Methode im Debugger-Fenster eine neue Registerkartengruppe.



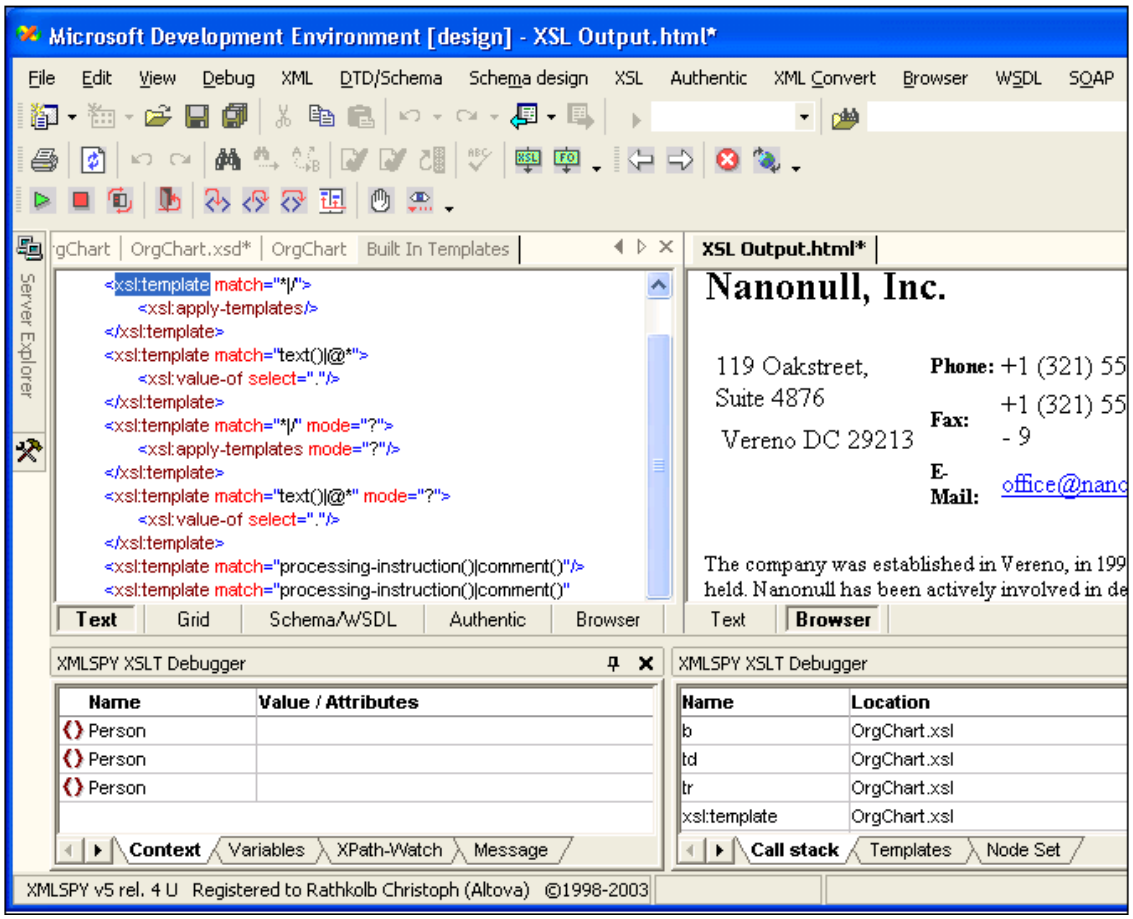
5. Ziehen Sie das ganz rechts angezeigte **XMLSPY XSLT Debugger**-Fenster nach rechts.
6. Wählen Sie im Popup-Menü "New Vertical Tab Group".



Sie können die Datei OrgChart.xml nun während des Debugging-Vorgangs im XSLT-Debugger

 © 2005 Altova GmbH

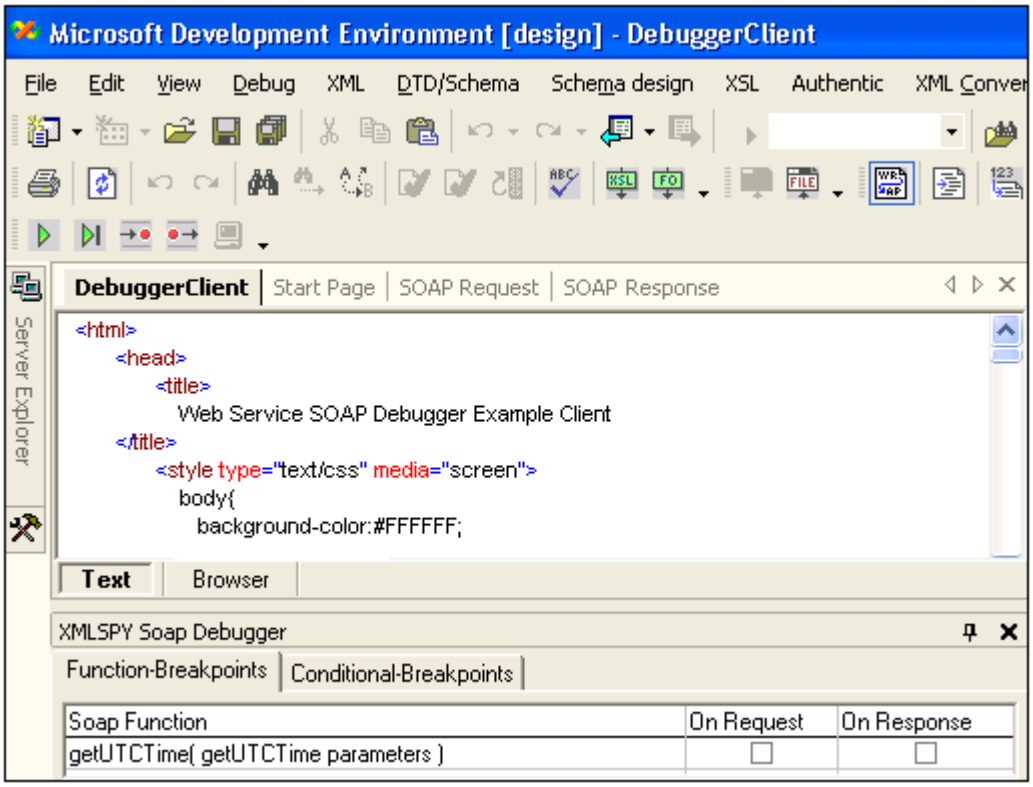
anzeigen:


Microsoft Development Environment [design] - XSL Output.html*
File Edit View Debug XML DTD/Schema Schema design XSL Authentic XML Convert Browser WSDL SOAP
gChart | OrgChart.xsd* | OrgChart | Built In Templates
XSL Output.html*
Server Explorer
<xsl:template match="*|/">
<xsl:apply-templates/>
</xsl:template>
<xsl:template match="text()|@*">
<xsl:value-of select="."/>
</xsl:template>
<xsl:template match="*|/" mode="?">
<xsl:apply-templates mode="?"/>
</xsl:template>
<xsl:template match="text()|@*" mode="?">
<xsl:value-of select="."/>
</xsl:template>
<xsl:template match="processing-instruction()|comment()"/>
<xsl:template match="processing-instruction()|comment()"
Nanonull, Inc.
119 Oakstreet,
Suite 4876
Vereno DC 29213
Phone: +1 (321) 55
Fax: +1 (321) 55 - 9
E-Mail: office@nanc
The company was established in Vereno, in 199
held. Nanonull has been actively involved in de
Text Grid Schema/WSDL Authentic Browser
Text Browser
XMLSPY XSLT Debugger
Name Value / Attributes
Person
Person
Person
Context Variables XPath-Watch Message
XMLSPY XSLT Debugger
Name Location
b OrgChart.xsl
td OrgChart.xsl
tr OrgChart.xsl
xsl:template OrgChart.xsl
Call stack Templates Node Set
XMLSPY v5 rel. 4 U Registered to Rathkolb Christoph (Altova) ©1998-2003

8.3 Visual Studio .NET und SOAP Debugger

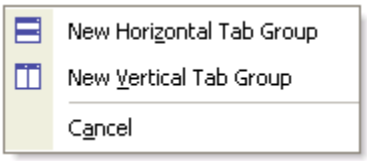
SOAP Debugger

Im unten gezeigten Screenshot sehen Sie, wie die Datei DebuggerClient.html angezeigt wird, wenn der SOAP Debugger aktiviert wurde. Standardmäßig werden alle SOAP-Fenster auf einer Registerkartengruppe angezeigt. Um den Debugging-Vorgang übersichtlicher zu gestalten, können Sie Ihre eigene Registerkartengruppe in Visual Studio .NET anlegen.



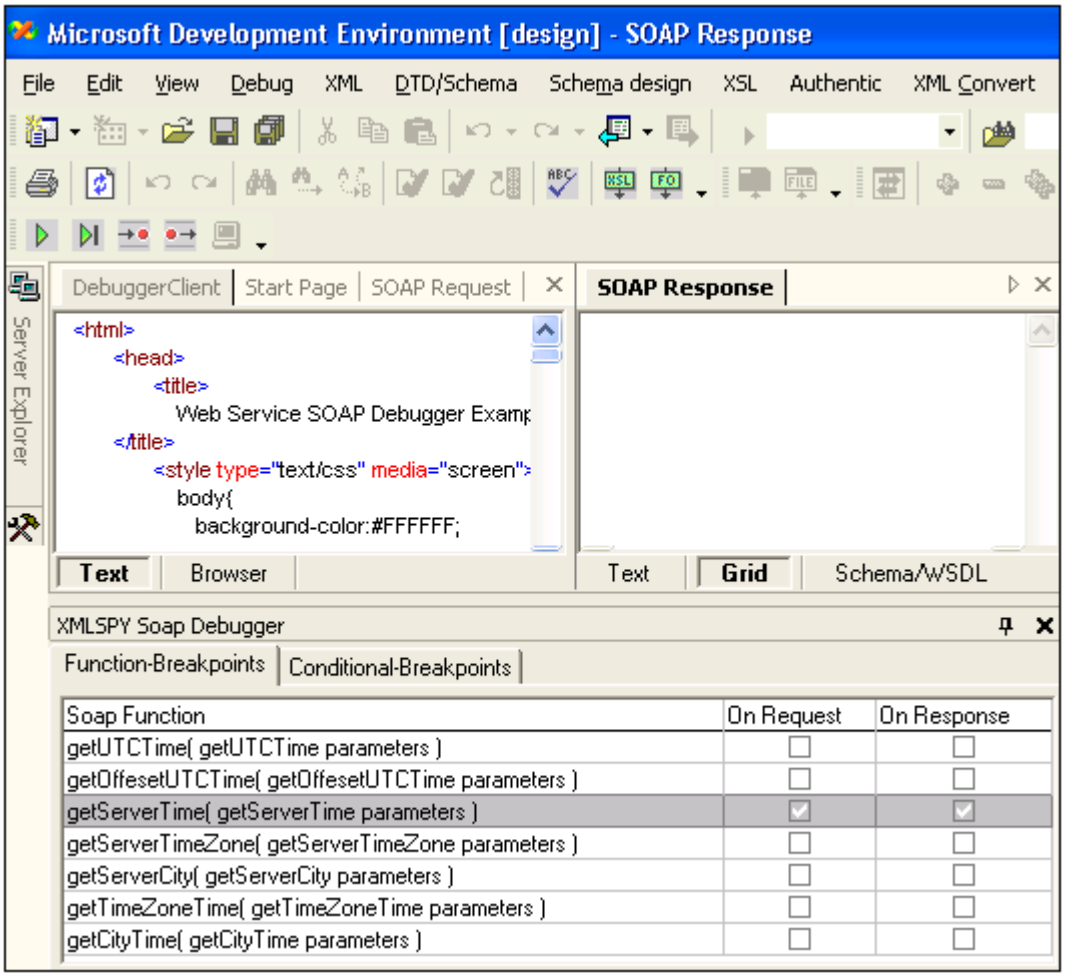
So legen Sie eine neue Registerkartengruppe an:

1. Klicken Sie auf das Register **SOAP Response** und ziehen Sie es auf die aktive Registerkarte.
2. Hierauf wird ein Popup-Menü angezeigt, in dem Sie die Art der zu erstellenden Registerkarte definieren können.



3. Wählen Sie die Option "New Vertical Tab Group".
   Daraufhin wird eine neue Registerkarte mit dem SOAP Response-Fenster erstellt.

© 2005 Altova GmbH


Microsoft Development Environment [design] - SOAP Response
File Edit View Debug XML DTD/Schema Schema design XSL Authentic XML Convert
DebuggerClient Start Page SOAP Request
SOAP Response
Server Explorer
<html>
<head>
<title>
Web Service SOAP Debugger Examp
</title>
<style type="text/css" media="screen">
body{
background-color:#FFFFFF;
Text Browser
Text Grid Schema/WSDL
XMLSPY Soap Debugger
Function-Breakpoints Conditional-Breakpoints
Soap Function
On Request
On Response
getUTCTime(getUTCTime parameters)
getOffesetUTCTime(getOffesetUTCTime parameters)
getServerTime(getServerTime parameters)
getServerTimeZone(getServerTimeZone parameters)
getServerCity(getServerCity parameters)
getTimeZoneTime(getTimeZoneTime parameters)
getCityTime(getCityTime parameters)

9 XMLSpy 2005 in Eclipse

Eclipse 3.0 ist eine Open-Source-Umgebung, in die verschiedene Arten von Applikationen in Form von Plug-Ins integriert werden können. XMLSpy 2005 für die Eclipse-Plattform ist ein Eclipse 3.0 Plug-In, das Ihnen gestattet, auf die Funktionalitäten einer zuvor installierten Version von XMLSpy 2005 Edition über die Eclipse 3.0-Plattform zuzugreifen.

Um das XMLSpy 2005 Plug-In für Eclipse 3.0 installieren zu können, benötigen Sie:

- die XMLSpy 2005 Edition, die Sie verwenden möchten: Enterprise, Professional oder Home
- das Eclipse 3.0-Paket sowie
- die entsprechende Java Runtime-Version

Laden Sie das XMLSpy Plugin für Eclipse vom Download-Bereich der Altova.com Website herunter. Während der Installation werden Sie aufgefordert, den Installationsordner für das Plug-In anzugeben.

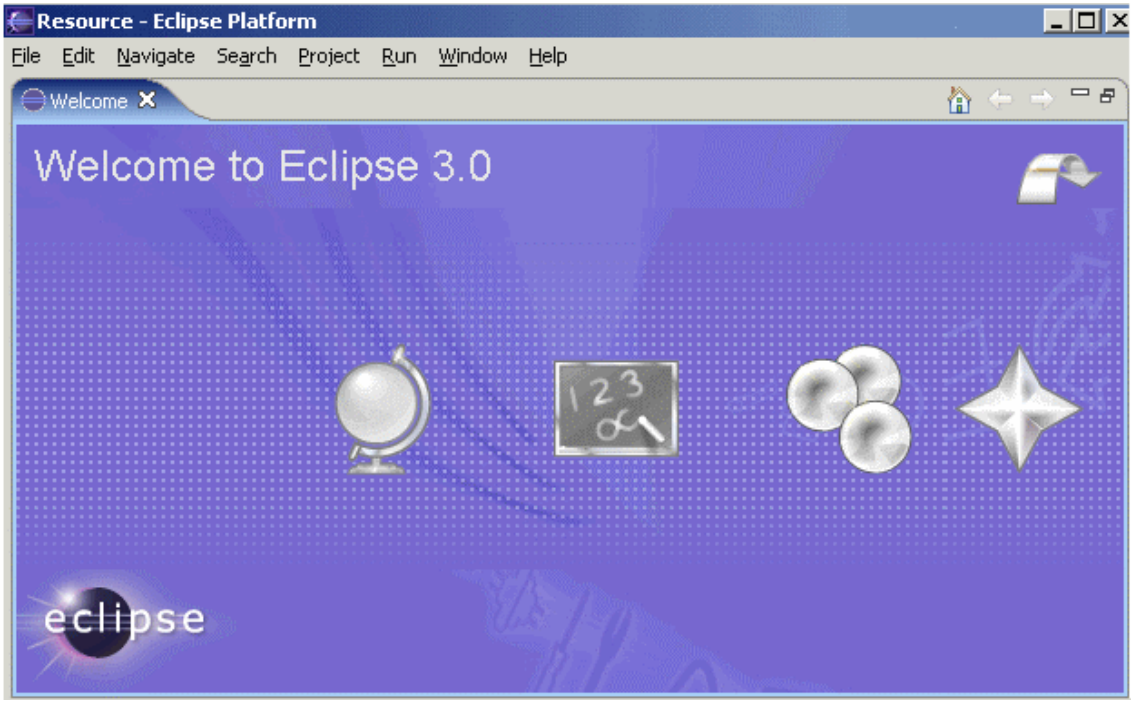
Das XMLSpy Plug-In für Eclipse bietet die folgenden Funktionalitäten:

- einen Editor mit vollem Funktionsumfang, mit dem Sie jede Dateiart, die XMLSpy 2005 unterstützt, bearbeiten können. Die applikationsspezifischen Befehle werden zur Menü- und Symbolleiste hinzugefügt.
- eine Reihe von Ansichten, die die einzelnen Anwendungsfenster enthalten: in diesem Fall die XMLSpy 2005 Eingabehilfen sowie die einzelnen spezifischen Debuggerfenster.
- verschiedene Perspektiven, die das Aussehen der Workbench bestimmen. XMLSpy 2005 bietet drei Perspektiven: XMLSpy, Debug SOAP und Debug XSLT.
- eine XMLSpy 2005 Benutzerhilfe unter dem Menüeintrag **Help | XMLSPY |** Inhaltsverzeichnis.

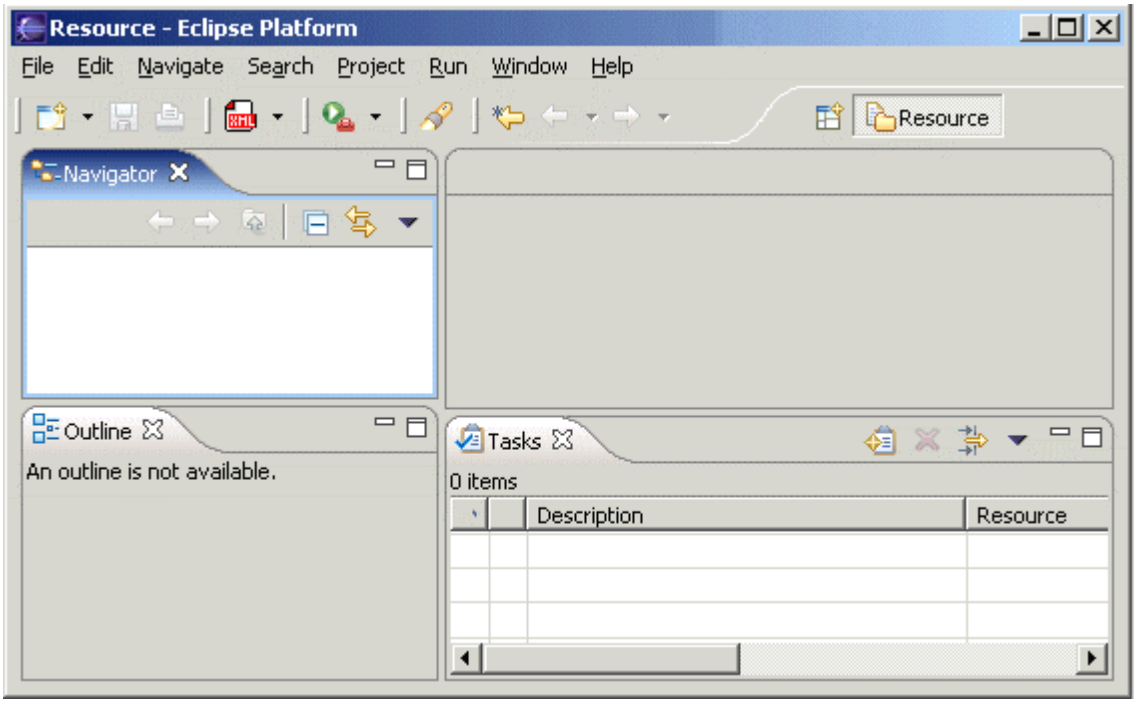
© 2005 Altova GmbH

9.1 Starten von Eclipse und Verwendung des XMLSpy 2005 Plug-Ins

1. Doppelklicken Sie auf eclipse.exe, um die Eclipse-Plattform zu starten.

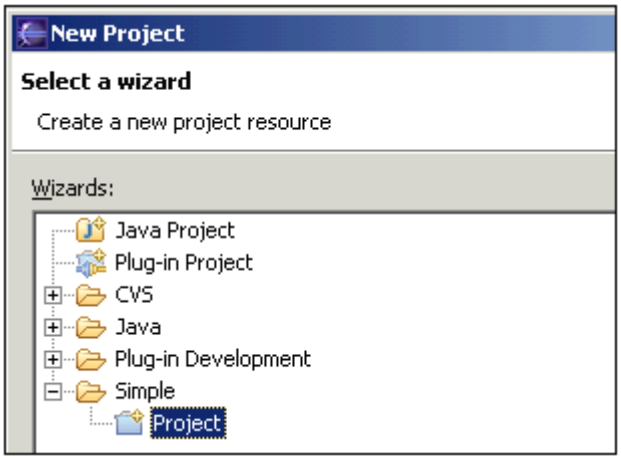


   Daraufhin erscheint der Willkommensbildschirm von Eclipse 3.0
2. Platzieren Sie den Cursor über das Pfeilsymbol und klicken Sie, wenn der Text "Go To Workbench" angezeigt wird.
3. Daraufhin wird ein leeres XMLSpy 2005-Fenster in Eclipse geöffnet.

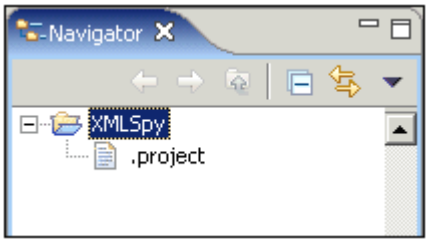


Erstellen eines neuen Projekts:

1. Rechtsklicken Sie in das Navigatorfenster und wählen Sie **New | Project | Simple Project**.



2. Geben Sie XMLSpy als Projektnamen ein und klicken Sie auf "Finish".
3. Daraufhin wird der XMLSpy-Projektordner erstellt.

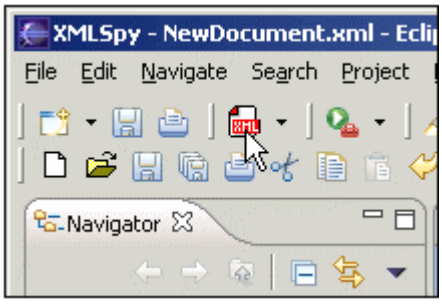


© 2005 Altova GmbH

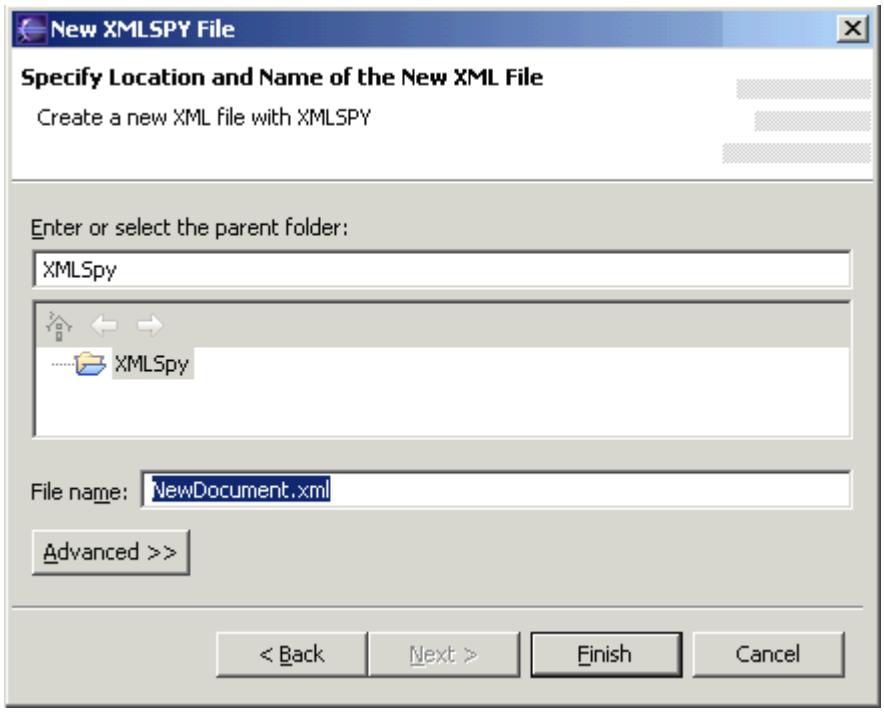
9.2 Erstellen von XML-Dateien

Erstellen einer neuen XML-Datei auf Basis eines Schemas

1. Klicken Sie auf das XMLSpy 2005 XML-Symbol.

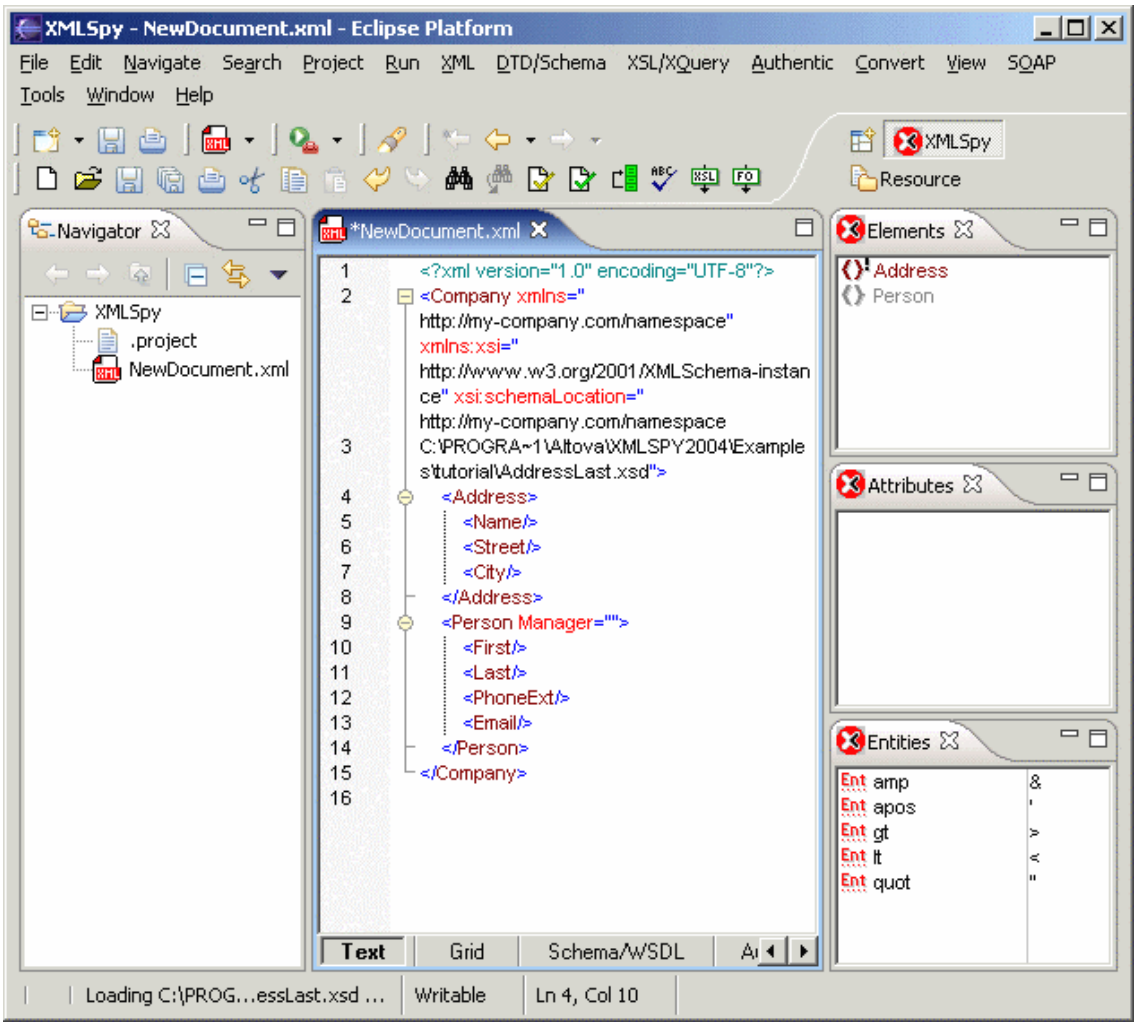


2. Wählen Sie **xml | XM-Dokument** und klicken Sie auf "Next".
   Sie werden nun aufgefordert, einen übergeordneten Ordner auszuwählen.



3. Geben Sie XMLSpy als übergeordneten Ordner (Parent folder) an (oder wählen Sie einen vorhandenen Ordner aus), verwenden Sie den vorgeschlagenen Standard-Dateinamen **NeuesDokument.xml** und klicken Sie auf "Finish".
   Die Dialogfelder, die nun angezeigt werden, stammen aus XMLSpy 2005.
4. Wählen Sie Schema und klicken Sie auf OK, wählen Sie anschließend mit Hilfe der Durchsuchen-Schaltfläche die Schemadatei (z.B. AddressLast.xsd) und klicken Sie auf OK.
   Die neue XML-Datei wird nun in der Textansicht auf dem Register NeuesDokument.xml angezeigt.

   Die vorkonfigurierte XMLSpy 2005-Perspektive wird automatisch aktiviert, um die verschiedenen Eingabehilfen anzuzeigen.

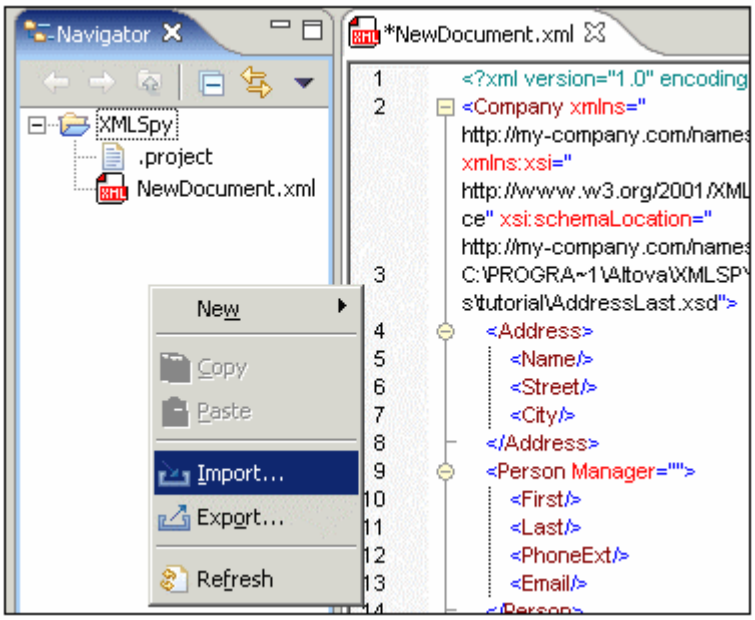


 © 2005 Altova GmbH

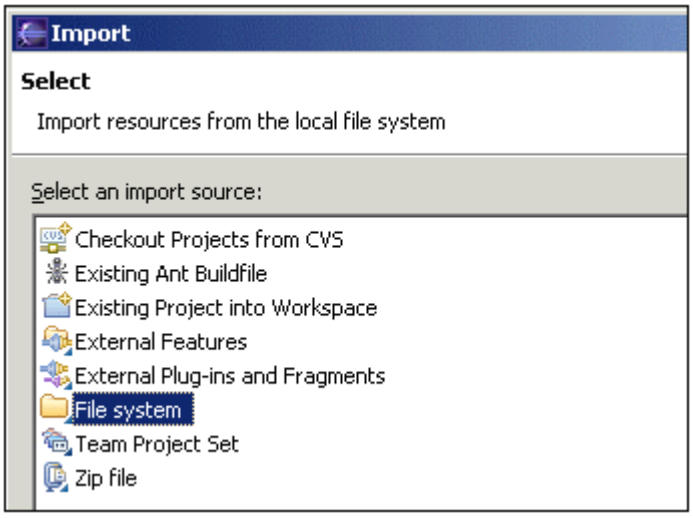
9.3 Importieren von XML-Dateien in Eclipse-Ordner

Importieren des XMLSpy 2005 Ordners "Examples" in den Navigator:

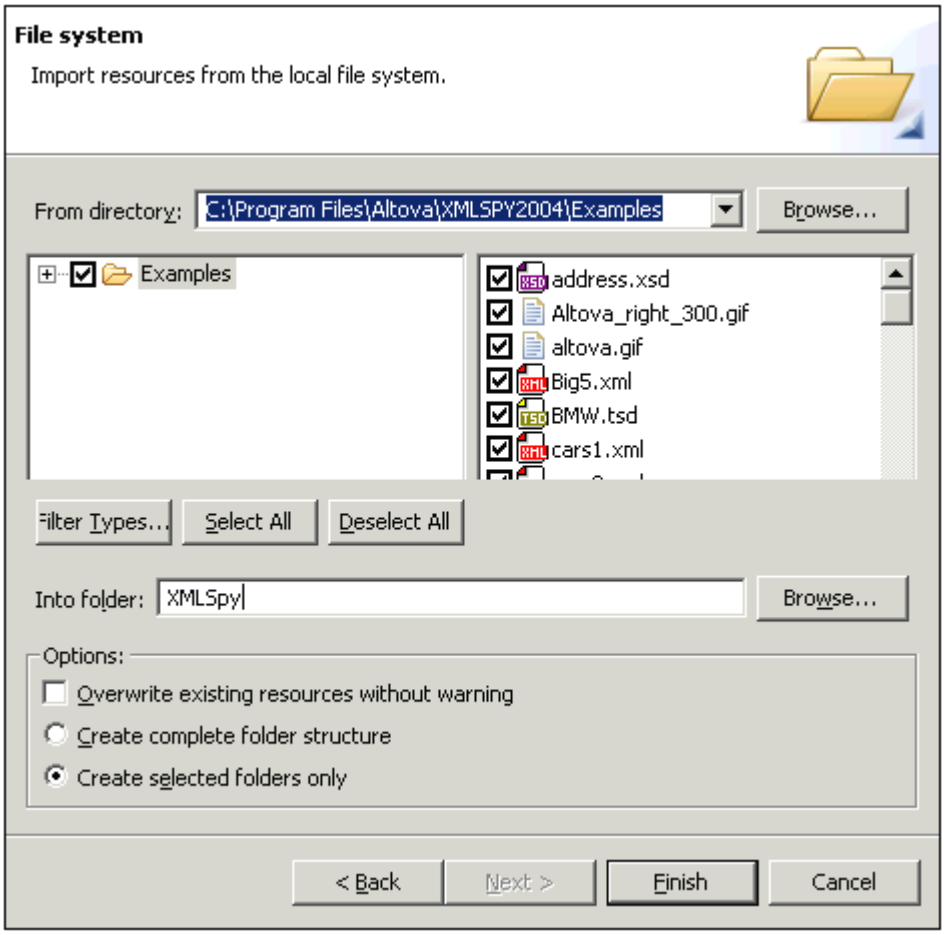
1. Rechtsklicken Sie auf das Register "Navigator" und wählen Sie "Import".



2. Wählen Sie "File system" und klicken Sie auf "Next".

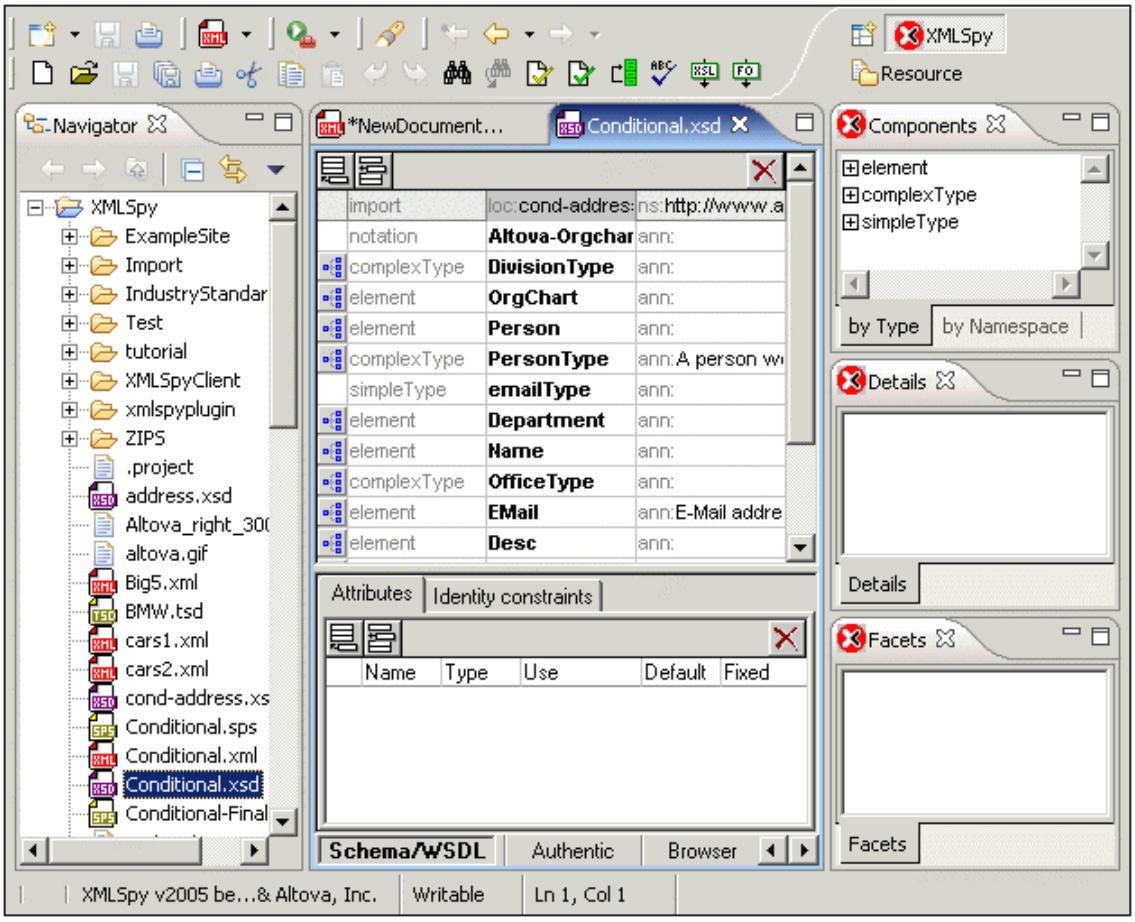


3. Klicken Sie auf die Schaltfläche "Browse" rechts vom Textfeld "From directory:" und wählen Sie das Verzeichnis "Examples" in Ihrem XMLSpy 2005-Ordner aus.



4. Aktivieren Sie das Kontrollkästchen **Examples**.
   Daraufhin werden alle Dateien in den verschiedenen Unterverzeichnissen im Fenster auf der rechten Seite aktiviert.
5. Klicken Sie auf die Schaltfläche "Browse" neben dem Textfeld "Into folder:" und wählen Sie den Zielordner aus. Klicken Sie anschließend auf "Finish".
   Die ausgewählten Ordner und Dateien werden in den Eclipse-Arbeitsbereich kopiert.
6. Doppelklicken Sie auf eine Datei im Navigator, um sie zu öffnen (z.B. Conditional.xsd).

© 2005 Altova GmbH


XMLSpy
Resource
Navigator
*NewDocument...
Conditional.xsd
Components
element
complexType
simpleType
by Type
by Namespace
Details
Facets
XMLSpy
ExampleSite
Import
IndustryStandar
Test
tutorial
XMLSpyClient
xmlspyplugin
ZIPS
.project
address.xsd
altova.gif
Big5.xml
BMW.tsd
cars1.xml
cars2.xml
cond-address.xs
Conditional.sps
Conditional.xml
Conditional.xsd
Conditional-Final
import
notation
complexType
element
element
complexType
simpleType
element
element
complexType
element
element
DivisionType
OrgChart
Person
PersonType
emailType
Department
Name
OfficeType
EMail
Desc
Attributes
Identity constraints
Name
Type
Use
Default
Fixed
Schema/WSDL
Authentic
Browser
XMLSpy v2005 be...& Altova, Inc.
Writable
Ln 1, Col 1

9.4 Unterschiede zwischen Eclipse und den Standalone-Versionen

Die Enterprise, Professional und Home Edition des Eclipse Plug-Ins für XMLSpy 2005 haben im Allgemeinen dieselben Funktionalitäten wie die jeweiligen Standalone-Versionen.

Nicht unterstützte Funktionen in den integrierten Versionen:

Info-Fenster
Das Info-Fenster wird nicht unterstützt. Dieses Fenster enthält zusätzliche Informationen zu ausgewählten Elementen/Attributen usw., d.h. es enthält den Namen des Elements/Attributs sowie Informationen zu dataype, enumeration und occurrence.

Projekt-Fenster und **Projekt-**Menü (sowie die Versionskontroll-Untermenüs).
Das Projekt-Menü (und die Versionskontroll-Untermenüs) stehen nicht zur Verfügung. Verwenden Sie stattdessen den Eclipse Navigator.

Extras | Skripting-Umgebung...
Die Skripting-Umgebung steht derzeit nicht zur Verfügung.

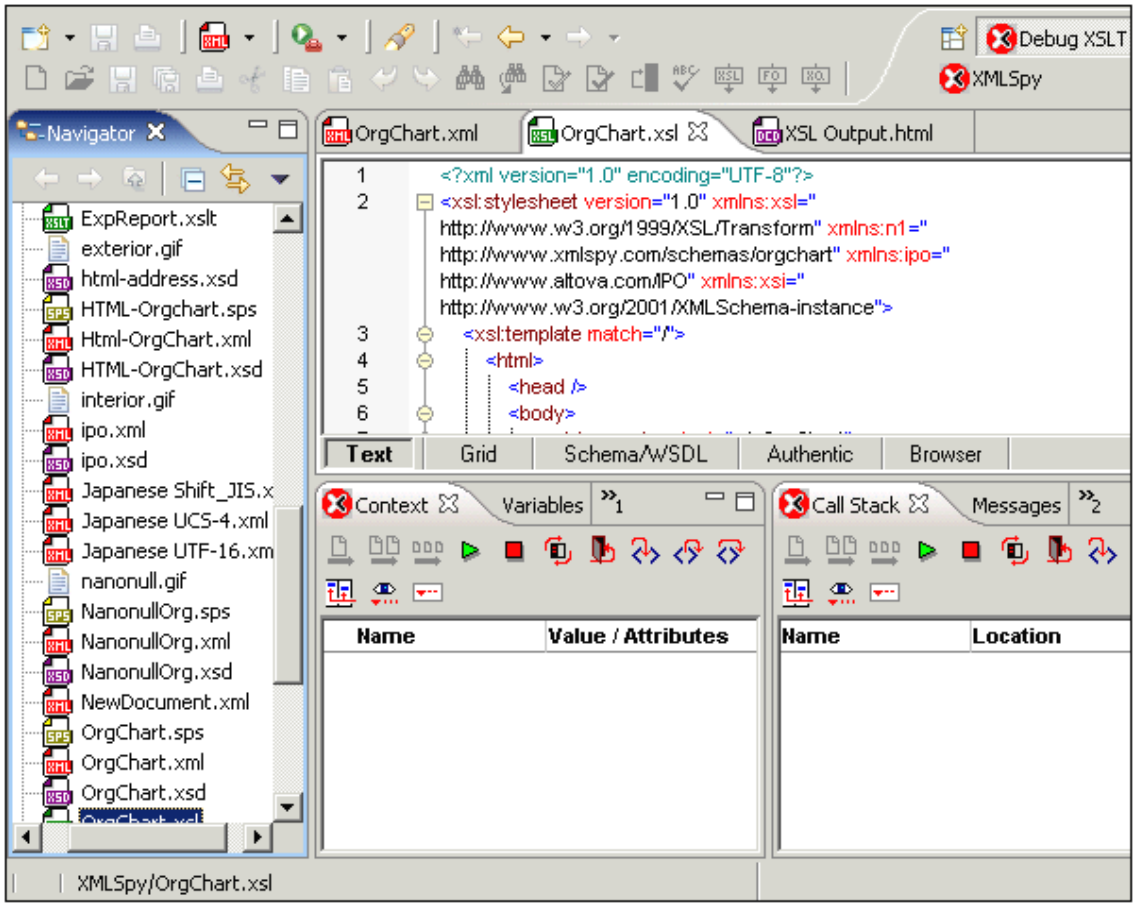
Eigenes Browser-Fenster
Das Kontrollkästchen "In eigenem Fenster anzeigen" im Menü **Extras | Optionen |** Register **Ansicht** wird nicht unterstützt, daher werden die Text- und die Browser-Ansicht immer in ein und demselben Fenster angezeigt - z.B. beim Transformieren einer XML-Datei in eine HTML-Datei.

Authentic-Ansicht
Symbole zum Textstatus sind in der Authentic-Ansicht nicht verfügbar. Die Funktionen bzw. die Formatierung, die Sie damit aktivieren, können jedoch weiterhin über die Eingabehilfe oder das Kontextmenü aufgerufen werden. Nähere Informationen dazu finden Sie in der Dokumentation zu Stylesheet Designer.

© 2005 Altova GmbH

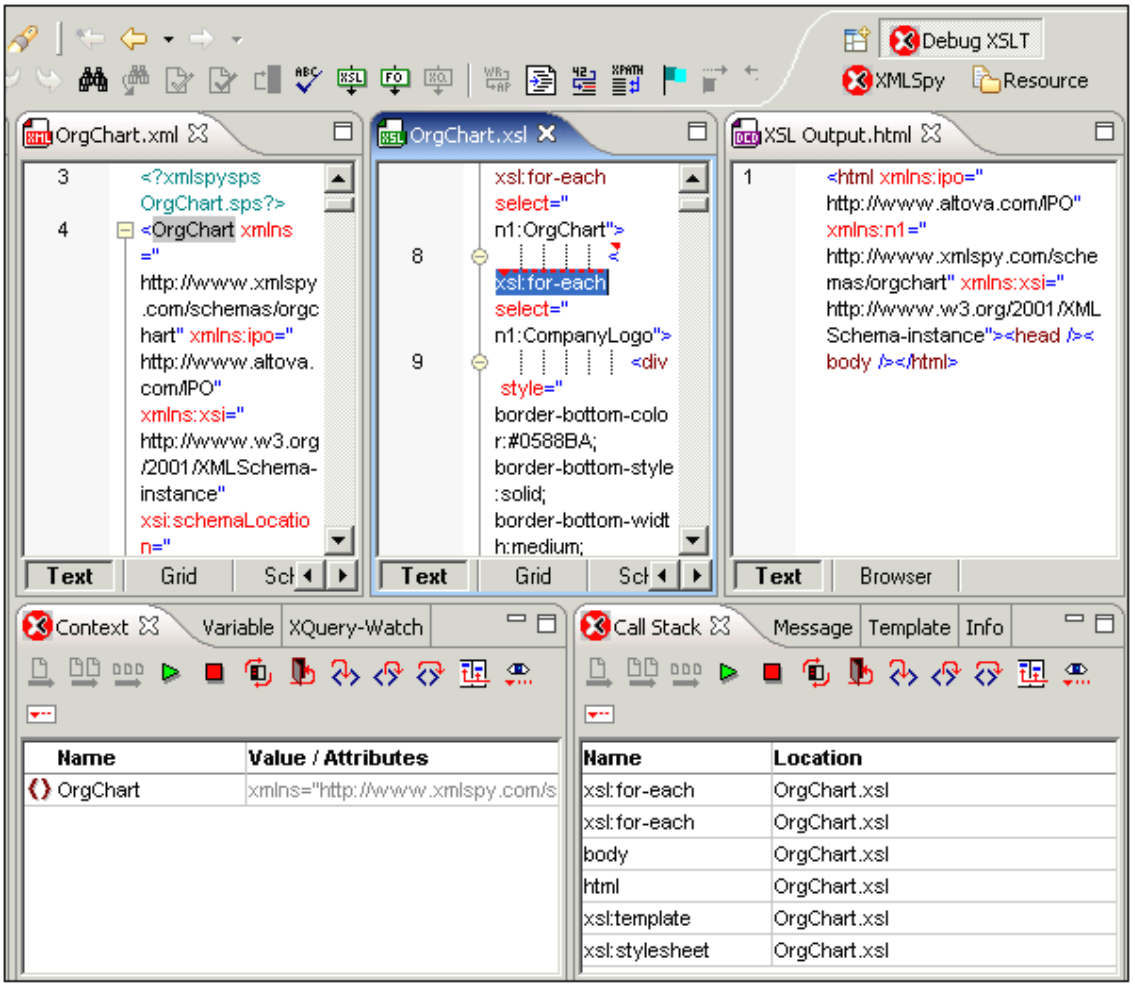
9.5 Eclipse-Plattform und XSLT Debugger

1. Öffnen Sie die Dateien **OrgChart.xml** und **Orgchart.xsl**.
2. Wählen Sie die Menüoption **XSL/XQuery | Debugger starten**.
3. Wählen Sie die gewünschte XML-Beispieldatei (klicken Sie auf die Schaltfläche "Fenster" und wählen Sie OrgChart.xml).
   Es erscheint eine Meldung, dass die Ansicht nun gewechselt wird. Bestätigen Sie mit OK.



Erstellen separater Dateifenster:

1. Klicken Sie auf das Register **XSL Output.html**, ziehen Sie es ganz nach rechts und lassen Sie die Maustaste los, wenn ein Pfeil erscheint. Dadurch wird das Register von der Registergruppe abgekoppelt.
   Beim Ziehen wird als Positionierungshilfe ein Fensterumriss angezeigt.
2. Entkoppeln Sie auch das Register **OrgChart.xsl** auf dieselbe Weise und positionieren Sie es zwischen die beiden Fenster.
   In der nachstehenden Grafik sehen Sie das neue Layout und den Debugging-Vorgang.

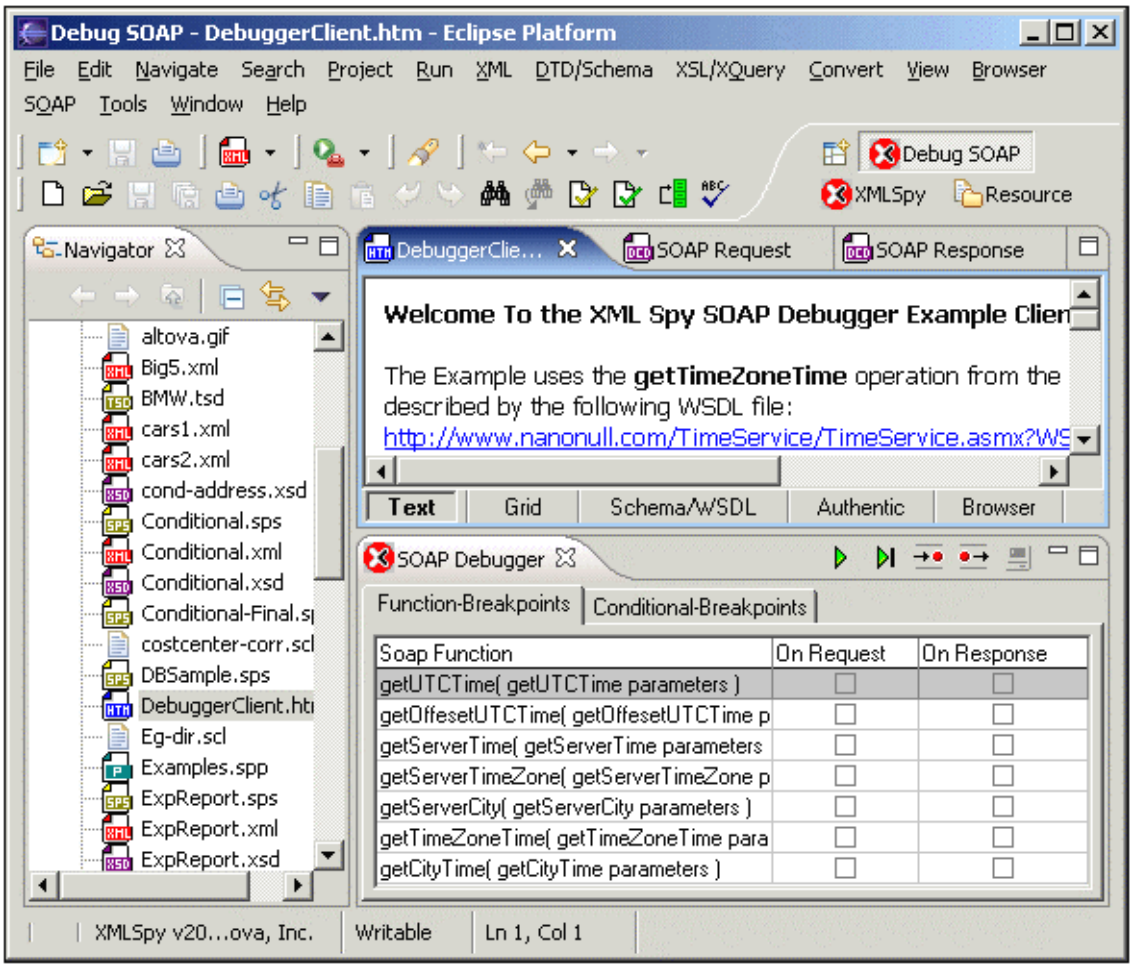


Wenn Sie die Debugger-Sitzung schließen, wird die XSL Output.html-Datei automatisch geschlossen und Sie wechseln in die XMLSpy 2005-Perspektive zurück.

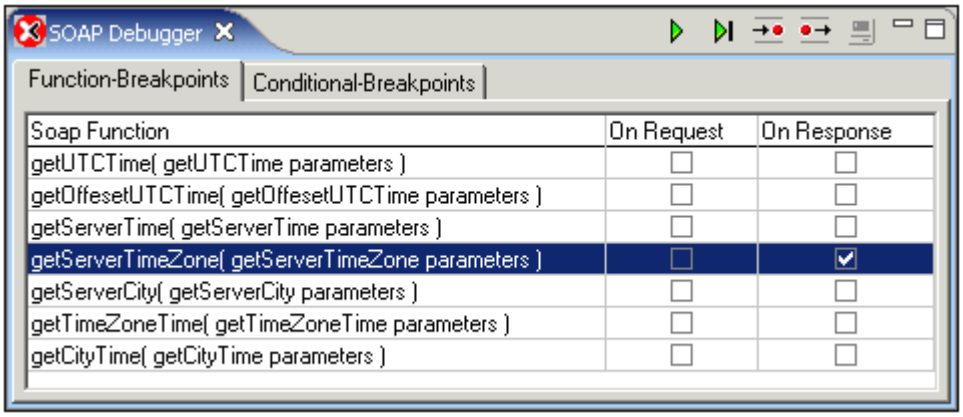
© 2005 Altova GmbH

9.6 Eclipse Plattform und SOAP Debugger

1. Öffnen Sie im Navigator die Datei **DebuggerClient.html**.
2. Wählen Sie die Menüoption **SOAP | SOAP Debugger-Sitzung**.
3. Wählen Sie im Ordner ..\Examples die Datei **TimeService.wsdl**.
4. Klicken Sie auf OK, um die Standard-Quell- und Zielports auszuwählen.

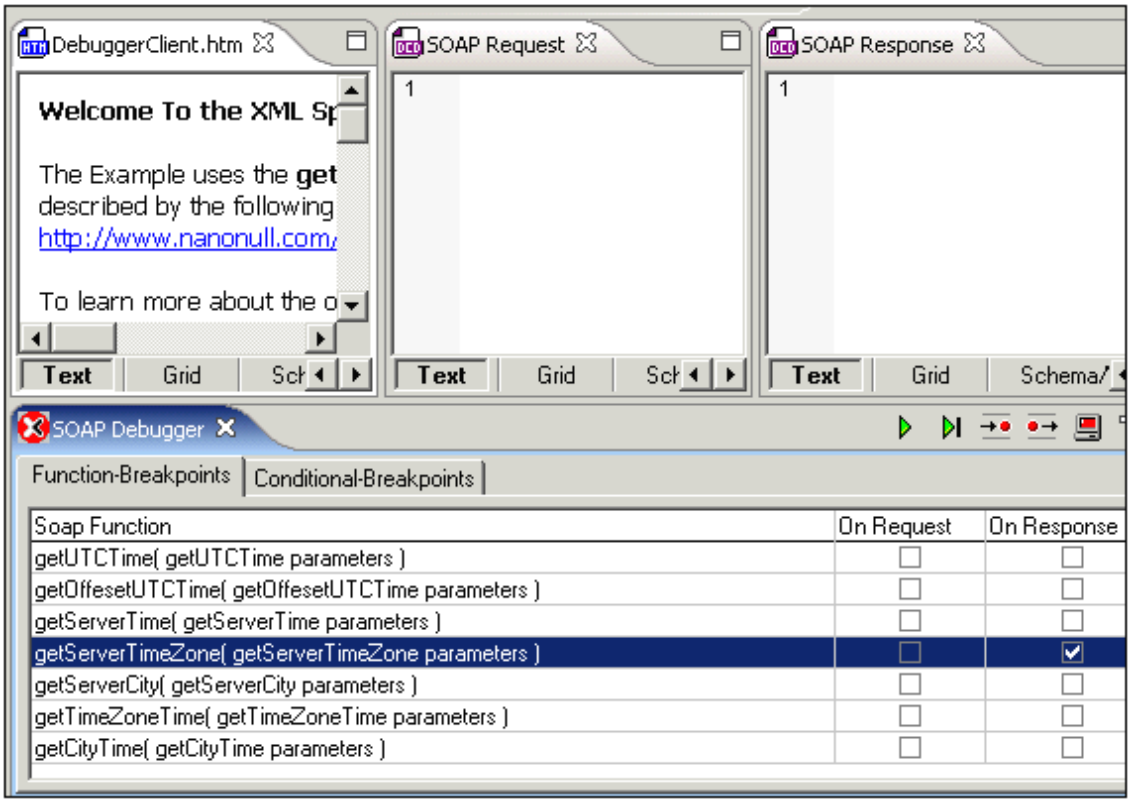


5. Setzen Sie auf dem SOAP Debugger-Registern einen Function Breakpoint (z.B. getServerTimeZone).



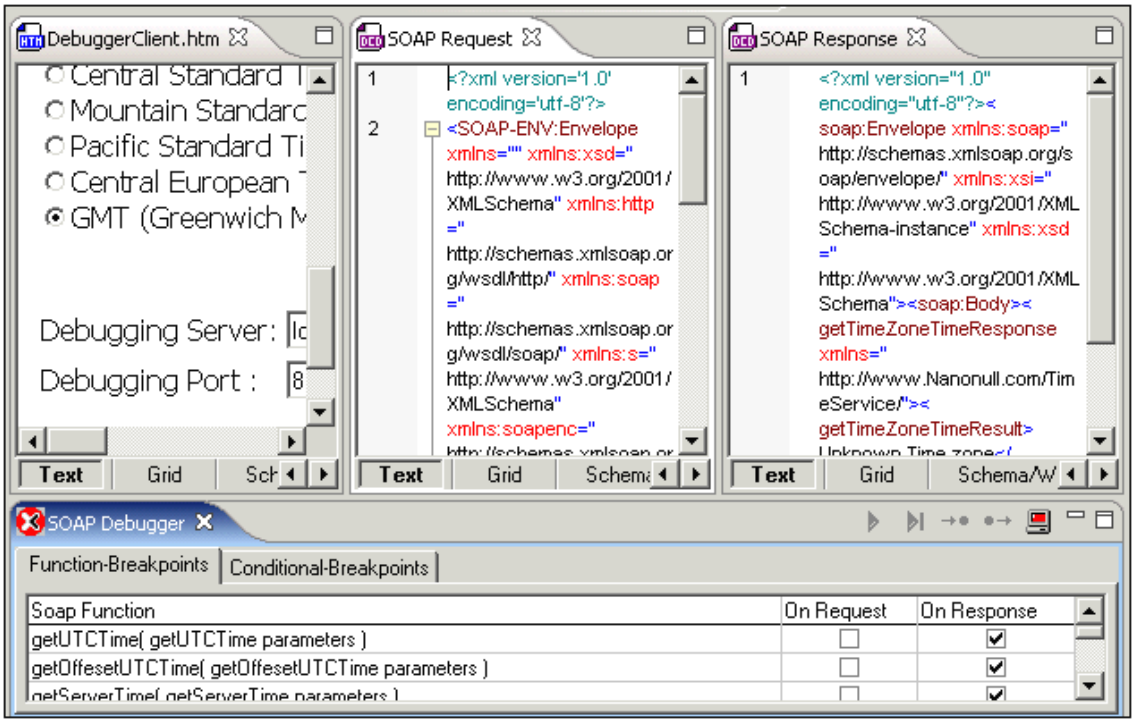
Erstellen separater Dateifenster:

1. Klicken Sie auf das Register "SOAP Response", ziehen Sie es ganz nach rechts und lassen Sie die Maustaste los, wenn ein Pfeil erscheint. Dadurch wird das Register von der Registergruppe abgekoppelt.
   Beim Ziehen wird als Positionierungshilfe ein Fensterumriss angezeigt.
2. Entkoppeln Sie auch das Register SOAP Request auf dieselbe Weise und positionieren Sie es zwischen die beiden Fenster



3. Wählen Sie die Menüoption **SOAP | Starten** oder klicken Sie auf das Symbol **Starten** im Debugger-Fenster.
4. Klicken Sie auf das Register **DebuggerClient.html,** scrollen Sie anschließend hinunter und klicken Sie auf die Schaltfläche **Turn on Debugging mode**.
5. Klicken Sie auf das Symbol "Beim nächsten Request anhalten" (oder auf das Optionsfeld GMT in der HTML-Datei) um den SOAP Request abzusenden.

© 2005 Altova GmbH

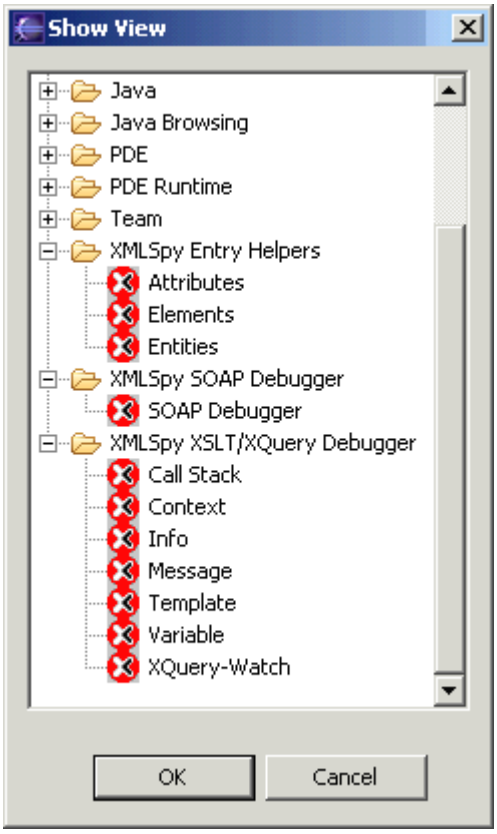


Wenn Sie die Debugger-Sitzung beenden, werden die Fenster SOAP Request und SOAP Response automatisch geschlossen und Sie wechseln in die XMLSpy 2005-Perspektive zurück.

9.7 Eclipse-Ansichten und -Perspektiven

Die XMLSpy 2005-Ansichten definieren die Haupt-Eingabehilfen sowie die Debugger-Fenster für die einzelnen Debugger.

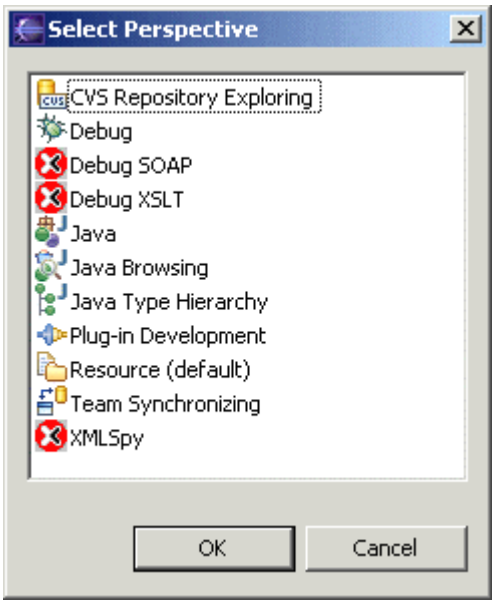
- Wählen Sie die Menüoption **Window | Show View | Other...,** um die aktuell verfügbaren Ansichten anzuzeigen.



Die XMLSpy 2005-Perspektiven definieren das Anwendungs-Hauptfenster sowie die Debugger-Oberflächen.

- Wählen Sie die Menüoption **Window | Open Perspective | Other...,** um die aktuell verfügbaren Perspektiven anzuzeigen.

© 2005 Altova GmbH


Select Perspective
CVS Repository Exploring
Debug
Debug SOAP
Debug XSLT
Java
Java Browsing
Java Type Hierarchy
Plug-in Development
Resource (default)
Team Synchronizing
XMLSpy
OK
Cancel

10 XSLT und XQuery Debugger

Der XSLT/XQuery Debugger dient zum Testen und Debuggen von XSLT Stylesheets und XQuery-Dokumenten. Über die Oberfläche des XSLT/XQuery Debugger können Sie gleichzeitig eine Ansicht des XSLT/XQuery-Dokuments, des Ergebnisdokuments und des XML-Quelldokuments anzeigen. Sie können sich anschließend Schritt für Schritt durch das XML-XSLT/XQuery-Dokument bewegen. Bei jedem Schritt wird die entsprechende Ausgabe generiert. Wenn eine XML-Quelldatei angezeigt wird, wird die entsprechende Position in dieser Datei bei jedem Schritt markiert. Gleichzeitig zeigen Fenster der Benutzeroberfläche Informationen zur Behebung von Fehlern an.

Der XSLT/XQuery Debugger wird immer während einer **Debugger-Sitzung** geöffnet. Es gibt drei Arten von Debugger-Sitzungen:

- XSLT 1.0, wobei der integrierte Altova XSLT 1.0-Prozessor verwendet wird
- XSLT 2.0, wobei der integrierte Altova XSLT 2.0-Prozessor verwendet wird
- XQuery, wobei der integrierte Altova XQuery 1.0-Prozessor verwendet wird

Die Art der Debugger-Sitzung wird automatisch gewählt und hängt von der Art des Dokuments ab, von dem aus die Debugger-Sitzung gestartet wird (Dieses Dokument bezeichnen wir hier als das *aktive Dokument* oder die *aktive Datei.*). XSLT-Debugger-Sitzungen werden von XSLT-Dateien aus gestartet (Die Version hängt vom Wert des Attributs `version` des Elements `xsl:stylesheet` (oder `xsl:transform`) im XSLT Stylesheet ab (`"1.0"` für XSLT 1.0 und `"2.0"` für XSLT 2.0).). XQuery-Debugger-Sitzungen werden von XQuery-Dateien aus gestartet. Wenn die aktive Datei eine XML-Datei ist, hängt die Auswahl davon ab, ob Sie eine XSLT- oder eine XQuery-Datei auf die XML-Datei anwenden. In ersterem Fall hängt die Auswahl der Debugger-Art davon ab, ob es sich beim Stylesheet um ein XSLT 1.0- oder ein XSLT 2.0-Stylesheet handelt..

In der nachfolgenden Tabelle sehen Sie eine Zusammenfassung dieser Informationen.

Aktive Datei	**Verknüpfte Datei**	**Debugger-Sitzung**
XSLT 1.0	XML; (erforderlich)	XSLT 1.0 (mit dem integrierten Altova XSLT 1.0-Prozessor)
XSLT 2.0	XML; (erforderlich)	XSLT 2.0 (mit dem integrierten Altova XSLT 2.0-Prozessor)
XQuery	XML; (optional)	XQuery (mit dem integrierten Altova XQuery-Prozessor)
XML	XSLT 1.0, oder XSLT 2.0, oder XQuery; (erforderlich)	XSLT 1.0, XSLT 2.0 oder XQuery. XSLT 1.0 oder 2.0 je nach Wert des Attributs `version` des Elements `xsl:stylesheet` (oder `xsl:transform`) im XSLT Stylesheet.

Nähere Informationen zu den drei Altova-Prozessoren finden Sie in den folgenden Abschnitten:

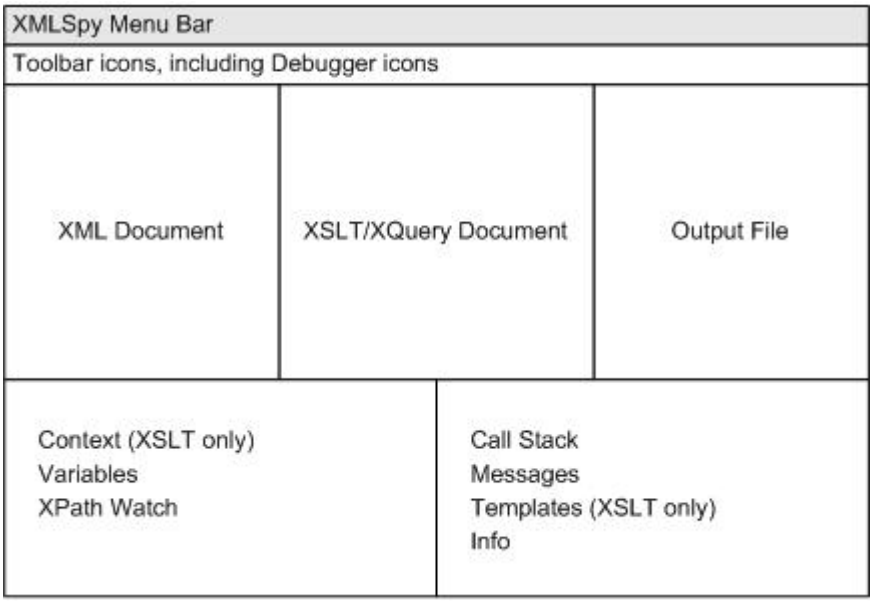
- Altova XSLT 1.0-Prozessor
- Altova XSLT 2.0-Prozessor
- Altova XQuery 1.0-Prozessor

© 2005 Altova GmbH

10.1 Funktionsweise und Benutzeroberfläche

Die grundlegende Vorgangsweise beim Debuggen von XSLT- und XQuery-Dateien mit XSLT/XQuery Debugger ist die folgende:

- Öffnen Sie eine Debugger-Sitzung. Die Art der Sitzung (XSLT 1.0, XSLT 2.0 oder XQuery) wird anhand der aktiven Datei ausgewählt (*siehe* XSLT und XQuery Debugger). Der XSLT/XQuery Debugger kann nur in der Textansicht und der Enhanced Grid-Ansicht verwendet werden. Wenn die Ansicht des aktiven Dokuments beim Start der Debugger-Sitzung nicht die Textansicht oder die Enhanced Grid-Ansicht ist, werden Sie gefragt, ob in die Textansicht, also die Standardansicht des XSLT/XQuery Debugger gewechselt werden darf. Sie können diese Option auch permanent aktivieren.
- Gehen Sie das XSLT- bzw. XQuery-Dokument schrittweise durch. Wenn eine XML-Datei mit der Sitzung verknüpft ist, wird die jeweilige Stelle in der XML-Datei markiert. Gleichzeitig wird für jeden Schritt in der Ergebnisdatei eine Ausgabe generiert, d.h. das Ergebnisdokument wird Schritt für Schritt erzeugt. Auf diese Art können Sie analysieren, welche Auswirkung jede einzelne Anweisung in der XSLT- bzw. XQuery-Datei hat.



Alternativ zur Anzeige aller drei Dokumente (XML, XSLT/XQuery, Ergebnisdokument) (siehe Abb. oben), können Sie auch eine Ansicht mit zwei Dokumenten (XSLT/XQuery und Ergebnisdokument) oder eine Ansicht mit nur einem einzigen Dokument auswählen.

- Während eine Debugger-Sitzung geöffnet ist, werden auf der Benutzeroberfläche Informationsfenster mit Informationen über die verschiedenen Aspekte der Transformation/Ausführung angezeigt (Variablen, XPath Watch, Call Stack, Messages, Info, usw.).
- Sie können den Debugger während einer Debugger-Sitzung anhalten (dies ist nicht dasselbe wie die Beendung der Debugger-Sitzung), um Änderungen am Dokument vorzunehmen. Alle Editierfunktionen, die in Ihrer XMLSpy 2005-Umgebung zur Verfügung stehen, können auch beim Editieren einer Datei während einer Debugger-Sitzung verwendet werden. Wenn der Debugger angehalten wird, bleibt die XSLT/XQuery Debugger-Oberfläche geöffnet und Sie haben Zugriff auf alle Informationen in den Debugger-Fenstern. Sie können nach dem Anhalten den Debugger in derselben Debugger-Sitzung erneut starten und das XSLT/XQuery-Dokument von Beginn an erneut debuggen.

- Sie können in der XSLT-Datei Breakpoints setzen, um die Verarbeitung an bestimmten Punkten zu unterbrechen. Damit wird der Vorgang beschleunigt, da Sie nicht jede einzelne Anweisung im XSLT- bzw. XQuery-Dokument manuell durchgehen müssen.
- Beenden einer Debugger-Sitzung. Dadurch wird die XSLT/XQuery Debugger-Oberfläche geschlossen und Sie kehren zur vorher angezeigten XMLSpy 2005 Umgebung zurück. Die Informationen in den Informationsfenstern stehen nun nicht mehr zur Verfügung. Die Informationen zu den Breakpoints bleiben jedoch in den jeweiligen Dateien erhalten, bis die Datei geschlossen wird. (Wenn Sie also eine weitere Debugger-Sitzung starten, bei der eine Datei mit Breakpoints verwendet wird, werden die Breakpoints in der neuen Debugger-Sitzung berücksichtigt.)

Hinweis: Die Debugger-Symbolleiste und -Symbole werden automatisch angezeigt, wenn eine Debugger-Sitzung gestartet wird.

© 2005 Altova GmbH

10.2 Befehle und Symbolleistenschaltflächen

Die Debugger-Befehle stehen im Menü **XSL/XQuery** und als Symbolleistenschaltflächen zur Verfügung. Die Debugger-Symbole erscheinen automatisch in der Symbolleiste, sobald eine Debugger-Sitzung geöffnet wird.

Debugger starten (Alt+F11)
Startet die Verarbeitung des XSLT/XQuery-Dokuments bzw. fährt damit fort, bis das Ende des Dokuments erreicht ist. Wurden Breakpoints gesetzt, so wird die Verarbeitung an diesen Stelle angehalten. Wenn der Debugger noch nicht gestartet wurde, starten Sie mit diesem Symbol die Sitzung, bis sie am ersten Node angehalten wird. Bei laufender Debugger-Sitzung wird das XSLT/XQuery-Dokument bis zu Ende verarbeitet oder bis der nächste Breakpoint erreicht wird.

Nur aktives Dokument ansehen
Maximiert das Fenster des gerade aktiven Dokuments auf der Debugger-Oberfläche.

XSLT/XQuery und Ausgabe ansehen
Zeigt das XSLT- und das Ausgabedokument an und blendet das XML-Dokument aus.

XML, XSLT/XQuery und Ausgabe anzeigen
Zeigt das XML-, XSLT/XQuery- und das Ausgabedokument an. Dies ist die Standardansicht, wenn ein XML-Dokument für die Debugger-Sitzung verknüpft wurde.

Debugger anhalten
Hält den Debugger an. Dies ist nicht dasselbe wie das Beenden der Debugger-Sitzung, in der der Debugger ausgeführt wird. Diese Funktion eignet sich dazu, um ein Dokument während der Debugger-Sitzung zu bearbeiten oder innerhalb derselben Debugger-Sitzung andere Dateien zu verwenden. Nachdem Sie den Debugger angehalten haben, müssen Sie ihn erneut starten, um am Anfang des XSLT/XQuery-Dokuments mit dem Debuggen zu beginnen.

Einsteigen (F11)
Geht die einzelnen Nodes und XPath-Ausdrücke im Stylesheet schrittweise durch. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem er angehalten wurde.

Überspringen (Strg+F11)
Überspringt den aktuellen Node und geht zum nächsten Node auf derselben Ebene bzw. zum

nächsten Node auf der nächsthöheren Ebene. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem er angehalten wurde.

Aussteigen (Umschalt+F11)
Steigt aus dem aktuellen Node aus und geht zum nächsten Geschwisternode des Parent-Node bzw. zum nächsthöheren Node oberhalb des Parent-Node.

Aktuellen Execution Node einblenden
Zeigt den aktuellen Execution Node im XSLT/XQuery-Dokument und den entsprechenden Kontext-Node im XML-Dokument an bzw. wählt diese aus. Dies ist nützlich, wenn Sie auf andere Register geklickt haben, die bestimmten Code im XSLT Stylesheet bzw. der XML-Datei anzeigen oder markieren, und zum Ausgangsort zurückkehren wollen.

Debugger neu starten
Löscht das Ausgabefenster und startet die Debugger-Sitzung mit den aktuell ausgewählten Dateien neu.

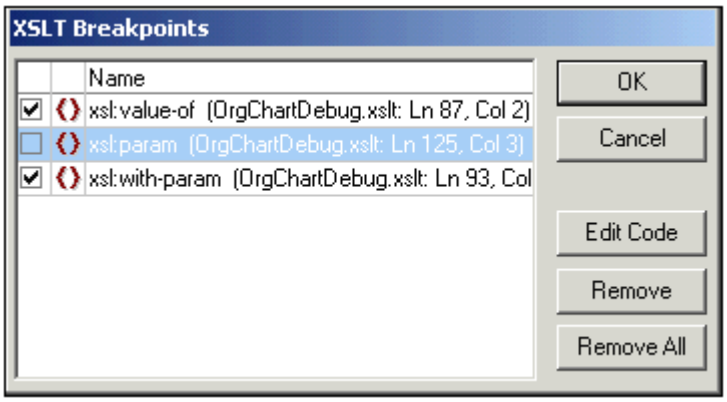
Breakpoint einfügen/löschen (F9)
Fügt einen Breakpoint an der aktuellen Cursorposition ein bzw. löscht diesen. Breakpoints innerhalb einer Zeile können für Nodes sowohl im XSLT/XQuery-Dokument als auch im XML-Dokument definiert werden. Sie legen fest, an welcher Stelle die Verarbeitung angehalten werden soll. Oberhalb des Nodes, an dem Sie einen Breakpoint gesetzt haben, erscheint eine strichlierte rote Linie. Für Abschlussnodes können keine Breakpoints gesetzt werden. Breakpoints in Attributen werden in XSLT-Dokumenten ignoriert. Der Befehl kann auch durch Rechtsklick auf die gewünschte Stelle aufgerufen werden.

Breakpoint ein/aus (Strg+F9)
Dieser Befehl (in der Symbolleiste nicht verfügbar) schaltet bereits definierte Breakpoints ein oder aus. Ein Breakpoint, der in aktiviertem Zustand rot angezeigt wird, erscheint grau, wenn er deaktiviert ist. Der Debugger hält an deaktivierten Breakpoints nicht an. Um einen Breakpoint ein- bzw. auszuschalten, setzen Sie den Cursor in den jeweiligen Node-Namen und klicken Sie auf den Befehl "Breakpoint ein/aus". Der Befehl kann auch durch Rechtsklick auf die gewünschte Stelle aufgerufen werden.

Debugger-Sitzung beenden
Beendet die Debugger-Sitzung und kehrt zur normalen XMLSpy 2005-Ansicht zurück, die vor dem Start der Debugger-Sitzung aktiv war. Ob die Ausgabedokumente, die für die Debugger-Sitzung geöffnet wurden, weiterhin geöffnet bleiben, hängt von den Einstellungen ab, die Sie im Dialogfeld XSLT/XQuery Debugger-Einstellungen vorgenommen haben.

© 2005 Altova GmbH

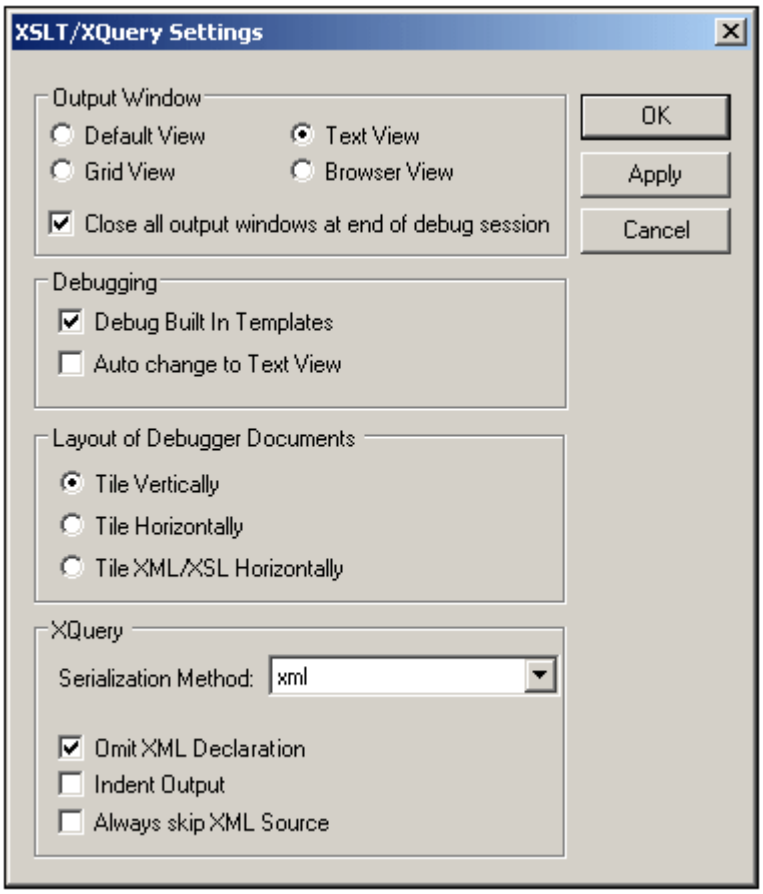
Dialogfeld "XSLT Breakpoints"
Dieser Befehl ruft das Dialogfeld "XSLT/XQuery Breakpoints" auf, in dem eine Liste aller in den Dateien der aktiven Debugger-Sitzung definierten Breakpoints (einschließlich aller deaktivierten Breakpoints) angezeigt wird.



Die Kontrollkästchen zeigen an, ob ein Breakpoint aktiviert (mit Häkchen versehen) oder deaktiviert ist. Sie können den markierten Breakpoint durch Klicken auf die Schaltfläche **Löschen** entfernen. Bei Klicken auf die Schaltfläche **Alle löschen** werden alle Breakpoints gelöscht. Bei Auswahl der Schaltfläche **Code bearbeiten** wird der entsprechende Breakpoint in der Datei direkt angezeigt.

10.3 Einstellungen

Im Dialogfeld XSLT/XQuery Debugger-Einstellungen können Sie Debugger- und Ausgabeoptionen einstellen, die für alle Debugger-Sitzungen gelten. Um das Einstellungsdialogfeld aufzurufen, wählen Sie den Befehl **XSL/XQuery | XSLT/XQuery-Einstellungen** oder klicken Sie auf das Symbol in der Symbolleiste. Die verschiedenen Einstellungen werden weiter unten beschrieben.



Ausgabefenster
Konfiguriert die Ansicht des Ausgabefensters (Standardansicht, Textansicht, Grid-Ansicht oder Browser-Ansicht). Die Standardansicht ist die Ansicht, die im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" für den Ausgabedateityp ausgewählt wurde. Für XSLT-Transformationen ist der Ausgabedateityp in der XSLT-Datei definiert. Für XQuery-Ausführungen ist der Ausgabedateityp durch die im Dialogfeld "XQuery-Einstellungen" ausgewählte "Serialization Method" festgelegt. (*siehe unten*).

Die Option "Alle Ausgabefenster nach Debuggen schließen" gibt Ihnen die Möglichkeit, die Ausgabefenster, die während der Debugger-Sitzung geöffnet waren, auch nach dem Schließen der Debugger-Sitzung geöffnet zu lassen.

Debugging
Bei Auswahl der Einstellung "Built-in Templates debuggen" **steigt** der Debugger, in built-in Templates **ein**, wann immer dies notwendig ist. Dies hat nichts mit der **Anzeige** von built-in Templates zu tun, wenn Sie auf dem Register "Templates" auf diese Art von Template-Eintrag klicken oder wenn der Callstack einen Node aus der Built-in Template-Datei anzeigt.

© 2005 Altova GmbH

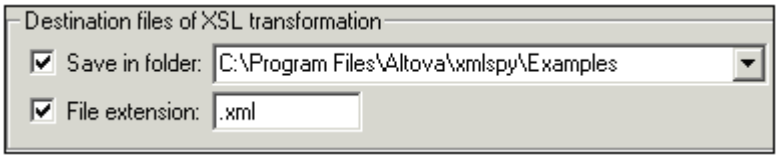
Der XSLT Debugger kann nur in der Text- oder der Grid-Ansicht verwendet werden. Bei Aktivierung der Option "Automatisch zu Textansicht wechseln" wird zum Debuggen automatisch in die Textansicht eines Dokuments gewechselt, wenn ein Dokument nicht in der Text- oder Grid-Ansicht angezeigt wird. (Der XQuery Debugger kann nur in der Textansicht verwendet werden.)

Layout von Debugger-Dokumenten
Die Debugger-Dokumente sind die Dokumente, die im Debugger geöffnet sind. Sie können auswählen, ob diese Dokumente vertikal oder horizontal bzw. mit der Option "XML/XSL horizontal anordnen" angeordnet werden sollen, wobei das Ergebnisdokument im letzteren Fall vertikal neben dem XML- und dem XSLT-Dokument angeordnet wird.

XQuery
Die "Serialization Method" bestimmt zwei Dinge: (i) das Dateiformat der generierten Ausgabedatei und (ii) die Regeln, die für das Schreiben der Ausgabe in die Ausgabedatei gelten. Zur Verfügung stehen die Optionen: HTML, Text, XHTML und XML. Je nach gewählter Option wird eine leere Datei des gewählten Dateityps erstellt, wenn Sie den Debugger innerhalb einer Debugger-Sitzung starten. Der Dateityp ist deshalb von Bedeutung, weil dadurch die aktuell aktiven XMLSpy 2005-Funktionen für diesen Dateityp verfügbar werden (wie z.B. die Validierung für XML-Dateien).

Sie haben die Möglichkeit, die XML-Deklaration auszulassen und die Ausgabe einzurücken. Bei Aktivieren der Option "XML-Quelle immer überspringen" können Sie den optionalen Schritt zur Verknüpfung der XML-Datei immer überspringen, wenn Sie eine Debugger-Sitzung von einer XQuery-Datei aus starten.

Einstellung der Auswahl eines anderen Ausgabeformats
Im Dialogfeld "Projekteigenschaften" können Sie für XSLT-Transformationen Verknüpfungen für einen Ordner erstellen. Eine dieser Optionen ist die Auswahl des Zielordners für die XSLT-Transformation, für den Sie eine Dateierweiterung wählen können.



Der hier ausgewählte Dateityp bestimmt den Dateityp des Ausgabeformats für XML- oder XSLT-Dateien, die zu einem Projekt gehören, für das XSLT-Transformationseigenschaften definiert wurden.

10.4 Starten einer Debugger-Sitzung

Am einfachsten lässt sich eine Debugger-Sitzung von einer XSLT-, XQuery- oder XML-Datei aus starten. Wenn der aktiven Datei bereits die erforderliche Datei (siehe Tabelle der verknüpften Dateien) zugewiesen wurde, wird die Debugger-Sitzung sofort gestartet. Andernfalls werden Sie aufgefordert, die erforderliche Datei zuzuweisen. Da XQuery-Dateien eine Verknüpfung mit einer XML-Datei weder benötigen noch enthalten, können Sie einstellen, ob Sie bei jedem Start einer XQuery Debugger-Sitzung (von einer XQuery-Datei aus) aufgefordert werden sollen, eine optionale XML-Dateiverknüpfung anzugeben, oder nicht.

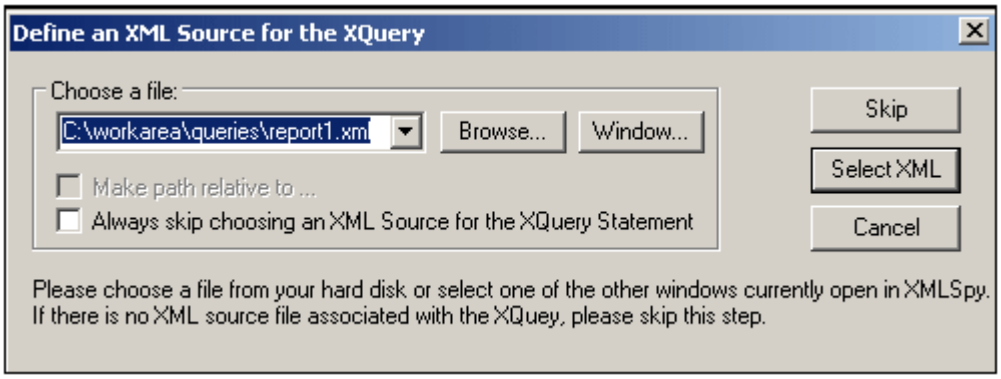
Vordefinierte Verknüpfungen
Vordefinierte Verknüpfungen sind nur für XSLT Debugger-Sitzungen relevant und beziehen sich auf Fälle, in denen die Dateiverknüpfung bereits in der aktiven Datei vorhanden ist. Um in einer XML- oder XSLT-Datei eine Zuweisung vorzunehmen, gehen Sie folgendermaßen vor:

- In XML-Dateien: Öffnen Sie die Datei, wählen Sie **XSL/XQuery | XSL zuweisen** und wählen Sie die XSLT-Datei aus.
- In XSLT-Dateien: Öffnen Sie die Datei, wählen Sie den Befehl **XSL/XQuery | XML-Beispieldatei zuweisen...** und wählen Sie die XML-Datei aus.

Wenn Sie den Befehl **XSL/XQuery | Debugger starten** wählen, wird die Debugger-Sitzung direkt gestartet, d.h. ohne dass Sie aufgefordert werden, die aktive Datei mit einer Datei zu verknüpfen.

Direkte Zuweisung
Falls die aktive Datei keine vordefinierte Verknüpfung enthält, werden Sie aufgefordert, eine anzugeben. Wenn Sie den Befehl **XSL/XQuery | Debugger starten** wählen, geschieht Folgendes:

- bei XML-Dateien: Sie werden aufgefordert, eine XSLT- oder XQuery-Datei auszuwählen.
- bei XSLT-Dateien: Sie werden aufgefordert, eine XML-Datei auszuwählen.
- bei XQuery-Dateien: Sie haben die Option, eine XML-Datei auszuwählen. Diese Option kann übersprungen werden.



(Das Dialogfeld in der Abbildung wird angezeigt, wenn Sie eine Debugger-Sitzung von einer XQuery-Datei aus starten.)

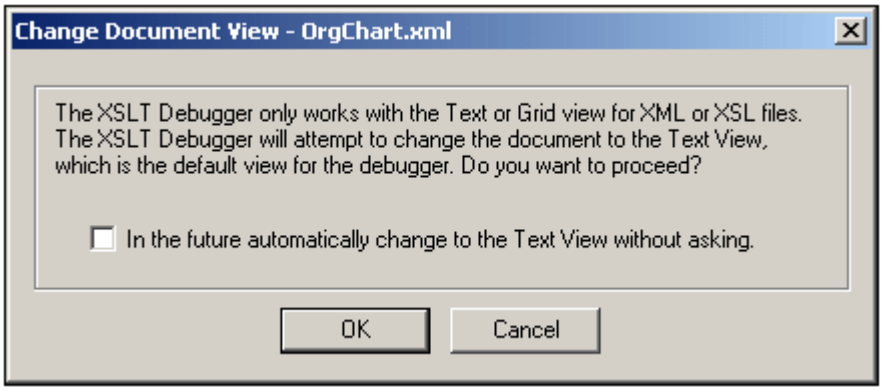
Nach Auswahl der erforderlichen Verknüpfungsdatei bzw. Auslassen der optionalen Verknüpfung wird die Debugger-Sitzung gestartet.

Alternative Methode zur Dateizuordnung

© 2005 Altova GmbH

Im Dialogfeld "Projekteigenschaften" können Sie Verknüpfungen vordefinieren. Klicken Sie auf **Projekt | Projekteigenschaften** und nehmen Sie die gewünschten Dateizuweisungen vor, indem Sie das Kontrollkästchen **XSL-Vorlage** / **XML-Vorlage** aktivieren.

Debugger-Ansicht
Der XSLT/XQuery Debugger funktioniert nur in der Text- und der Grid-Ansicht. Wenn die jeweilige XML- oder XSLT-Datei in einer anderen Ansicht geöffnet ist, bzw. wenn eine SPS-Datei mit einer XML-Datei verknüpft ist, erscheint das folgende Dialogfeld, wenn Sie eine Debugger-Sitzung unter Verwendung einer dieser Dateien starten.

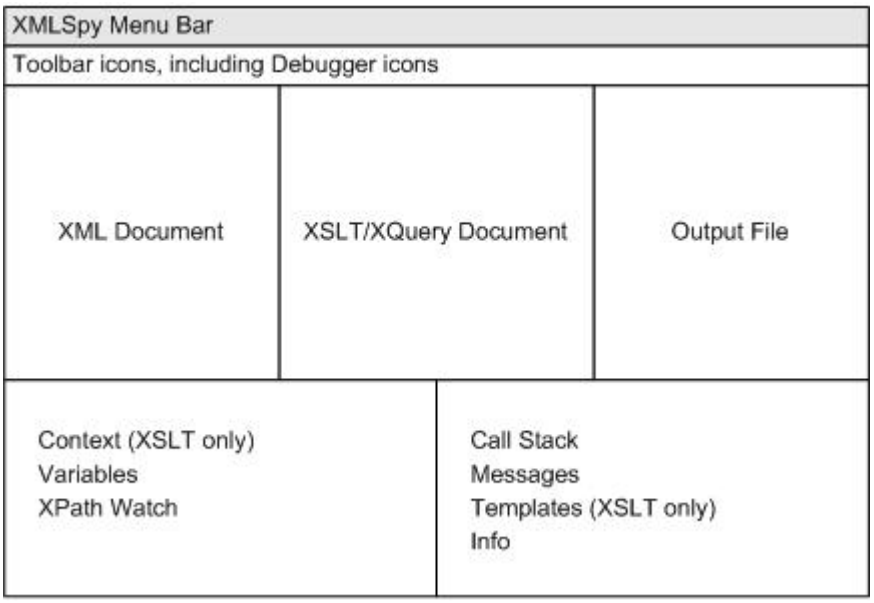


Wenn Sie auf OK klicken, wird das Dokument in der Textansicht geöffnet. Beachten Sie, dass XQuery-Dateien immer in der Textansicht angezeigt werden.

10.5 Informationsfenster

Informationsfenster, die während der Debugger-Sitzung im XSLT/XQuery Debugger-Anwendungsfenster geöffnet werden, enthalten Informationen über verschiedene Debugging-spezifische Aspekte der XSLT-Transformation bzw. der XQuery-Ausführung.

In XSLT-Debugger-Sitzungen stehen sieben Informationsfenster zur Verfügung, in XQuery-Debugger-Sitzungen fünf. Diese Fenster sind standardmäßig in zwei Gruppen unterteilt und sind am unteren Rand der XSLT/XQuery Debugger-Oberfläche (*siehe Abbildung unten*) angeordnet. Die Fenster und die darin angezeigten Informationen werden in diesem Abschnitt näher beschrieben.



Standard-Layout der XSLT/XQuery Debugger-Oberfläche.

In der ersten Informationsfenstergruppe werden die folgenden Fenster in Form von Registerkarten in einem einzigen Fenster angezeigt:

- Kontext (nur bei XSLT Debugger-Sitzungen)
- Variablen
- XPath-Watch

In der zweiten Informationsfenstergruppe werden die folgenden Fenster in Form von Registerkarten in einem einzigen Fenster angezeigt:

- Call Stack
- Messages
- Templates (nur bei XSLT Debugger-Sitzungen)
- Info

Im Standard-Layout gibt es zwei Fenstergruppen, jeweils mit Registern für die verschiedenen Fenster darin. Ein Register darin ist jeweils aktiv. Wenn Sie z.B. Informationen über Variablen anzeigen möchten, klicken Sie in der ersten Informationsfenstergruppe auf das Register "Variablen". Daraufhin wird das Variablenfenster angezeigt, während die Informationsfenster "Kontext" und "XPath-Watch" ausgeblendet werden. Beachten Sie, dass Sie auf einigen Registern die darin angezeigten Informationen auch zum Navigieren verwenden können: Wenn Sie auf einen Eintrag klicken, wird das jeweilige Element/Attribut in der XML-, XSLT- oder XQuery-Datei angezeigt. Nähere Informationen dazu finden Sie in der Dokumentation zu den jeweiligen Informationsfenstern (Kontext, Call Stack, Templates).

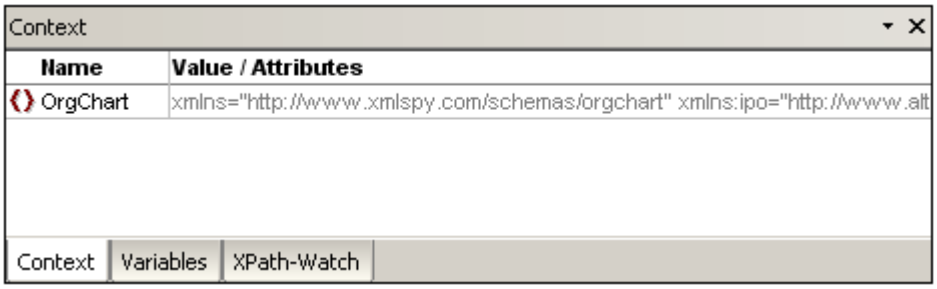
 © 2005 Altova GmbH

Durch Ziehen der Umrandung der beiden Informationsfenstergruppen können Sie deren Größe anpassen. Einzelne Fenster können aus der Gruppe herausgezogen werden, indem Sie auf den Registernamen klicken und das Fenster aus der Gruppe herausziehen. Um ein Fenster zur Gruppe hinzuzufügen, ziehen Sie seine Titelleiste auf die Titelleiste der Gruppe. Beachten Sie, dass es keine Schaltfläche zum Zurücksetzen auf das Standard-Layout gibt.

10.5.1 Kontext

Das Fenster "Kontext" steht nur in XSLT Debugger-Sitzungen nicht aber in XQuery Debugger-Sitzungen zur Verfügung.

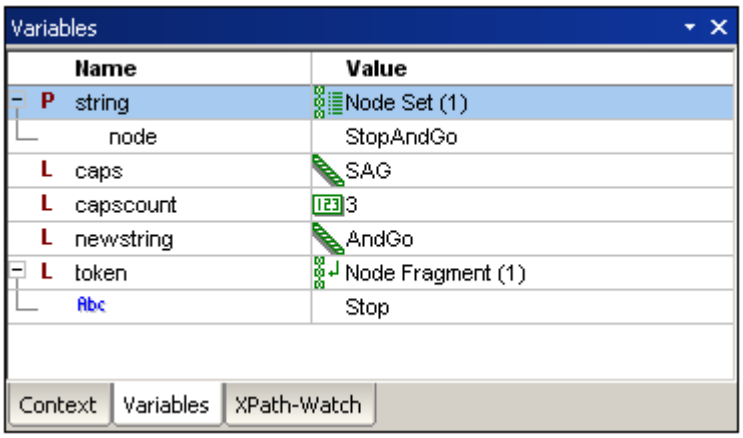
Während der Verarbeitung eines XSLT Stylesheet befindet sich der Prozessor-Kontext immer in einer Template, die mit einer Sequenz (von Nodes oder atomaren Werten) übereinstimmt. Im Kontextfenster wird der aktuelle Verarbeitungskontext angezeigt. Dabei kann es sich um eine Node-Sequenz, einen einzelnen Node oder einen atomaren Wert (wie z.B. einen String) handeln. Je nach Kontext-Eintrag, wird/werden der Wert bzw. das/die Attribut(e) angezeigt. Wenn es sich beim Kontext-Eintrag z.B. um ein Element handelt, werden die Attribute des Elements angezeigt. Wenn es sich beim Kontext-Eintrag um ein Attribut oder einen Text-Node handelt, wird der Wert des Node angezeigt.



Wenn Sie auf einen Eintrag im Kontextfenster klicken, wird diese Stelle im XML-Dokument angezeigt, Wenn das XML-Dokument derzeit auf der Benutzeroberfläche nicht angezeigt wird, wird ein Fenster dafür geöffnet.

10.5.2 Variablen

Das Fenster "Variablen" steht in XSLT und XQuery Debugger-Sitzungen zur Verfügung. Es werden darin die Variablen und Parameter und deren Werte angezeigt, die im XSLT/XQuery-Dokument verwendet werden, wenn Sie im Geltungsbereich liegen.

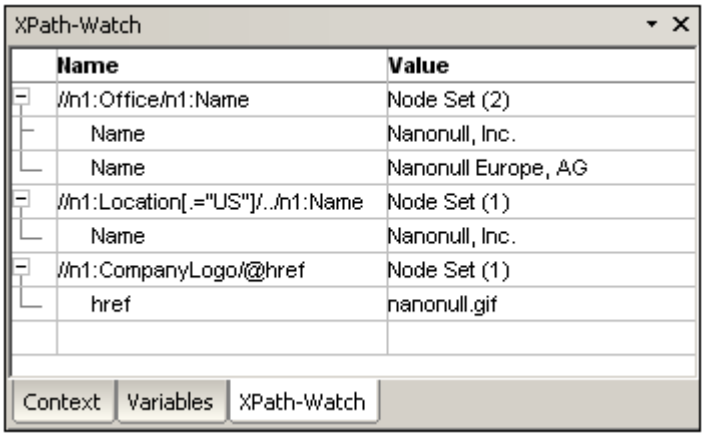


Parameter werden durch den Buchstaben **P** gekennzeichnet, globale Variablen (Variablen auf der obersten Ebene eines Stylesheet) durch ein **G** und lokale Variablen (innerhalb einer XSLT-Template deklarierte) durch ein **L**. Die Art des Werts von Variablen und Parameters wird im Feld "Value" ebenfalls durch Symbole gekennzeichnet. Es wird zwischen den folgenden Arten unterschieden: Node Set, Node Fragment, String, Number und Boolean.

10.5.3 XPath-Watch

Das Fenster "XPath-Watch" steht in XSLT- und XQuery-Debugger-Sitzungen zur Verfügung.

Sie können über dieses Fenster XPath-Ausdrücke eingeben, die Sie in einem oder mehreren Kontexten überprüfen möchten. Während Sie das XSLT-Dokument Schritt für Schritt überprüfen, wird der XPath-Ausdruck im aktuellen Kontext evaluiert und das Ergebnis wird in der Spalte "Value" angezeigt.



Um einen XPath-Ausdruck einzugeben, doppelklicken Sie in das Textfeld unterhalb der Spalte "Name" und geben Sie den XPath ein. Verwenden Sie Ausdrücke, die gemäß der XPath-Version, die der XSLT-Version des XSLT Stylesheet entspricht (XPath 1.0 für XSLT 1.0 und XPath 2.0 für XSLT 2.0), korrekt sind.

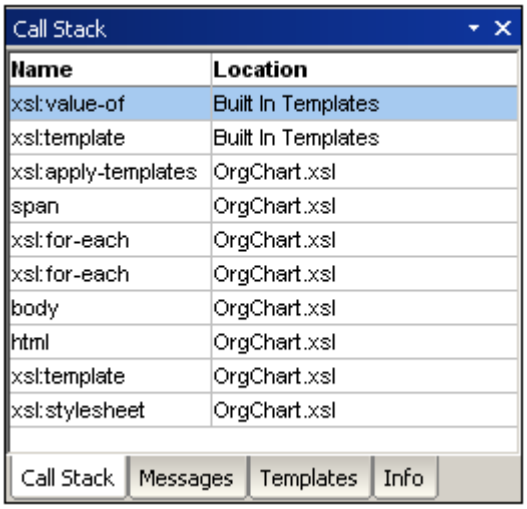
Hinweis: Wenn in der XML- bzw. XSLT-Datei Namespaces verwendet wurden, müssen Sie in Ihren XPath-Ausdrücken die richtigen Namespace-Präfixe verwenden.

10.5.4 Call Stack

Das Fenster "Call Stack" wird in XSLT- und XQuery Debugger-Sitzungen angezeigt.

Im Fenster "Call Stack" wird eine Liste aller zuvor verarbeiteten XSLT Templates und -Anweisungen angezeigt, wobei die aktuelle Template/Anweisung als erste in der Liste angezeigt wird.

© 2005 Altova GmbH

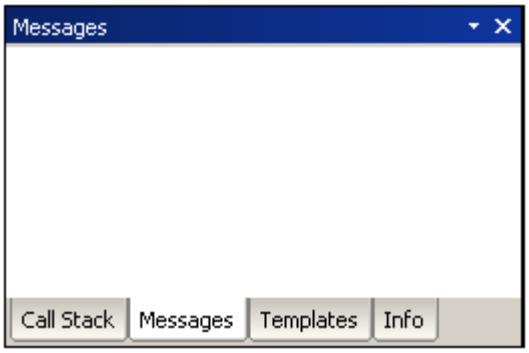


Wenn Sie in diesem Fenster auf einen Eintrag klicken, wird die jeweilige XSLT-Template/Anweisung im XSLT-Dokumentenfenster angezeigt. Wenn Sie auf eine Template/Anweisung klicken, die eine Built-in Template referenziert, wird die jeweilige built-in Template in einem separaten Fenster angezeigt, in dem alle Built-in Templates zu sehen sind.

10.5.5 Messages

Das Fenster "Messages" wird in XSLT- und XQuery-Debugger-Sitzungen angezeigt.

XSLT 1.0 und XSLT 2.0
In XSLT Debugger-Sitzungen werden auf dem Register "Messages" Fehlermeldungen, die `xsl:message`-Anweisung(en) oder andere Fehlermeldungen, die beim Debuggen auftreten können, angezeigt.



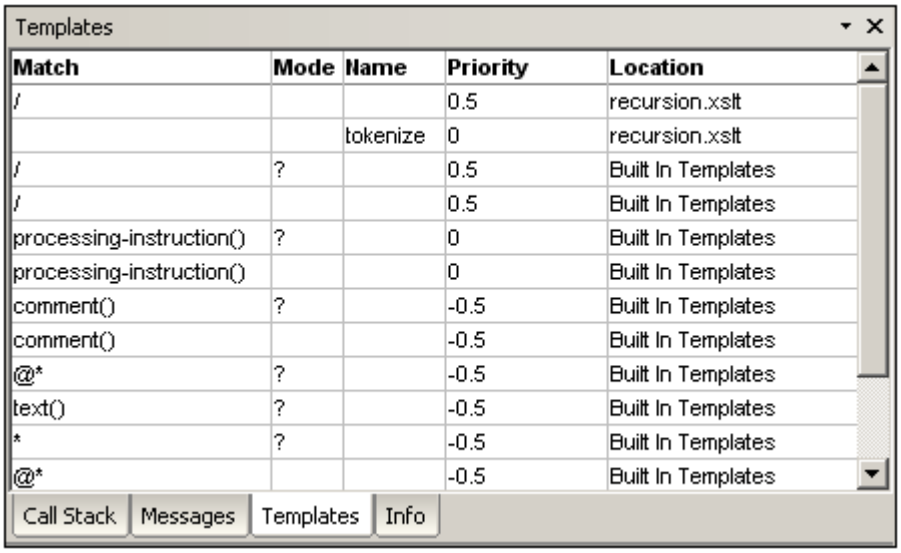
XQuery
In XQuery-Debugger-Sitzungen werden im Fenster "Messages" Fehlermeldungen angezeigt.

10.5.6 Templates

Das Fenster "Templates" (*siehe Abbildung unten*) ist nur in XSLT Debugger-Sitzungen verfügbar, nicht aber in XQuery Debugger-Sitzungen.

Im Fenster "Templates" werden die verschiedenen in XSLT Stylesheets verwendeten Templates angezeigt - einschließlich Built-in Templates und Named Templates. Matched

Templates werden nach den Nodes aufgelistet, mit denen diese übereinstimmen. Named Templates werden nach Namen aufgelistet. Für beide Template-Arten werden Mode (Modus), Priority (Priorität) und Location (Pfad) angezeigt.

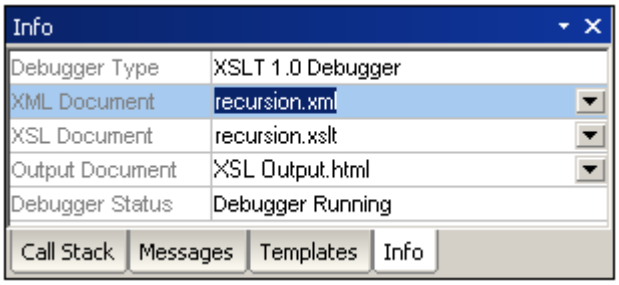


In oben gezeigten Screenshot sehen Sie zwei Templates im XSLT Stylesheet: Eine Template, die mit dem Dokument Node `/` übereinstimmt und eine Template mit dem Namen `tokenize`. Alle anderen Templates sind Built-in Templates (die im Feld "Location" durch keinen Eintrag gekennzeichnet sind).

Wenn Sie auf einen Eintrag in diesem Fenster klicken, wird die Template im XSLT-Dokumentenfenster markiert. Wenn Sie auf eine Built-in Template klicken, wird die Template in einem separaten Fenster, in dem alle Built-in Templates zu sehen sind, markiert.

10.5.7 Info

Das Fenster "Info" (*siehe Abbildung unten*) ist in XSLT- und XQuery Debugger-Sitzungen verfügbar und enthält Informationen über die aktuelle Debugger-Sitzung wie u.a. Informationen zur Version des verwendeten Debuggers, die Namen des Quell- und Zieldokuments und den Status des Debuggers.

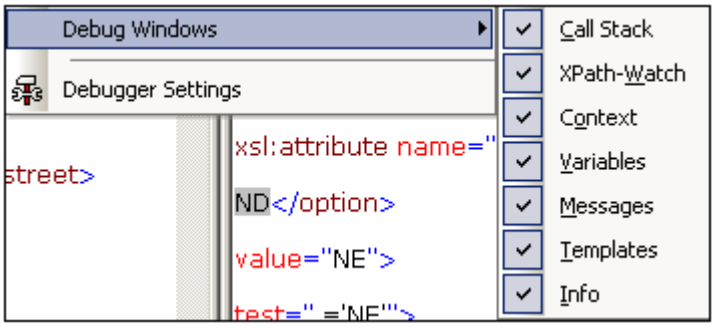


10.5.8 Anordnen der Informationsfenster

Die Informationsfenster können an verschiedenen Stellen des XSLT/XQuery Debugger-Anwendungsfensters angeordnet werden. Fenster können angedockt werden, freischwebend angeordnet oder als mehrere Fenster in einem Fenster angezeigt werden. Zum Anordnen der Fenster stehen Ihnen die folgenden Möglichkeiten zur Verfügung.

 © 2005 Altova GmbH

Menü
Platzieren Sie den Cursor im Menü "XSL/XQuery" über den Eintrag "Debug-Fenster", um eine Liste von Info-Fenstern anzuzeigen. Durch Klicken auf den entsprechenden Eintrag können Sie bestimmte Fenster ein- oder ausblenden.



Damit blenden Sie das Fenster ein- bzw. aus.

Kontextmenü
Das Kontextmenü wird durch Rechtsklick auf ein Register oder die Titelleiste eines Fensters aufgerufen.

Klicken Sie auf die gewünschte Option, um das Fenster an- oder abzudocken oder auszublenden.

Drag-and-Drop
Sie können ein Fenster an seinem Register oder der Titelleiste an die gewünschte Stelle ziehen.

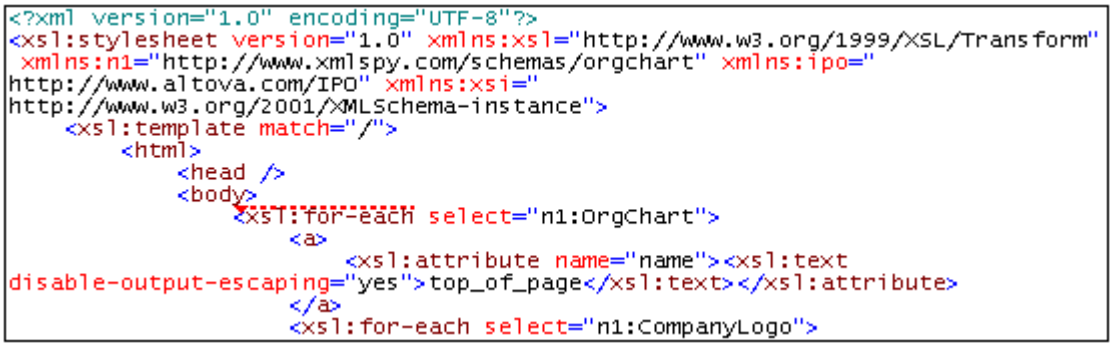
Außerdem können Sie das Fenster beim Ziehen mit Hilfe von Steuerelementen zur Platzierung an einem anderen Fenster oder im Anwendungsfenster andocken. Es gibt zwei Platzierungssteuerelemente: Wenn Sie ein Fenster über ein anderes Fenster ziehen, erscheint ein kreisförmiges Steuerelement, das aus fünf Sektoren besteht. Wenn Sie die Maustaste über einem dieser Sektoren loslassen, wird das Fenster, das Sie gerade gezogen haben, am entsprechenden Bereich des **Zielfensters** angedockt. Mit Hilfe der vier Pfeilsektoren wird das Fenster an der entsprechenden Seite des Zielfensters angedockt. Mit Hilfe der Schaltfläche in der Mitte wird das Fenster als Register des Zielfensters angedockt. Sie können ein Fenster auch als Register in einem anderen Fenster andocken, indem Sie es in die Registerleiste ziehen.

Die zweite Steuerelementgruppe erscheint beim Ziehen eines Fensters. Diese Gruppe besteht aus vier Pfeilen, einem für jede Seite der **Debugger-Oberfläche**. Wenn Sie die Maustaste beim Ziehen eines Fensters über einem dieser Pfeile loslassen, wird das Fenster an einer Seite der Debugger-Oberfläche angedockt.

Außerdem können Sie zwischen der an- und der abgedockten Position eines Fensters wechseln, indem Sie auf die Titelleiste des Fensters doppelklicken.

10.6 Breakpoints

Der XSLT/XQuery Debugger gestattet Ihnen, Breakpoints in XSLT-, XQuery- und XML-Dokumenten zu definieren. Breakpoints werden als strichlierte rote Linie angezeigt (*siehe Abb. unten*).



Wenn Sie den Debugger in einer Debugger-Sitzung starten, wird der Debugger an jedem Breakpoint angehalten. Auf diese Art können Sie bestimmte Bereiche, auf die Sie Ihre Aufmerksamkeit im XSLT-, XQuery- und/oder XML-Dokument richten möchten, für das Debuggen kennzeichnen. Sie können beliebig viele Breakpoints definieren.

Breakpoints in XML-Dokumenten
Sie können an jedem Node in einem XML-Dokument einen Breakpoint setzen. Die Verarbeitung wird am Beginn des Node angehalten.

Breakpoints in XSLT-Dokumenten
Sie können an den folgenden Stellen in einem XSLT-Dokument Breakpoints setzen:

- am Anfang von Templates und Template Instructions (z.B. `xsl:for-each`).
- an einem XPath-Ausdruck (XPath 1.0 oder XPath 2.0).
- an jedem Node in einem hartcodierten XML-Fragment. Die Verarbeitung wird am Beginn des Node unterbrochen.

Breakpoints in XQuery-Dokumenten
Sie können an den folgenden Stellen in einem XQuery-Dokument Breakpoints setzen:

- am Beginn von XQuery-Anweisungen.
- in einem XQuery-Ausdruck.
- an jedem Node in einem hartcodierten XML-Fragment. Die Verarbeitung wird am Beginn des Node unterbrochen.

Setzen von Breakpoints
So setzen Sie einen Breakpoint:

1. Setzen Sie den Cursor an die Stelle im Dokument, an der der Breakpoint gesetzt werden soll (*siehe Anmerkungen oben*). In XSLT Debugger-Sitzungen können Sie sowohl in der Text- als auch in der Grid-Ansicht Breakpoints setzen. XQuery-Debugger-Sitzungen stehen nur in der Textansicht zur Verfügung.
2. Fügen Sie den Breakpoint mit Hilfe einer der unten angeführten Methoden ein.

Einfügen von Breakpoints
Verwenden Sie eine der folgenden Methoden, um einen Breakpoint einzufügen:

© 2005 Altova GmbH

- Klicken Sie auf das Symbol .
- Wählen Sie **XSL/XQuery | Breakpoint einfügen/löschen**.
- Drücken Sie die Taste **F9**.
- Klicken Sie auf die rechte Maustaste und wählen Sie **Breakpoint einfügen/löschen**.

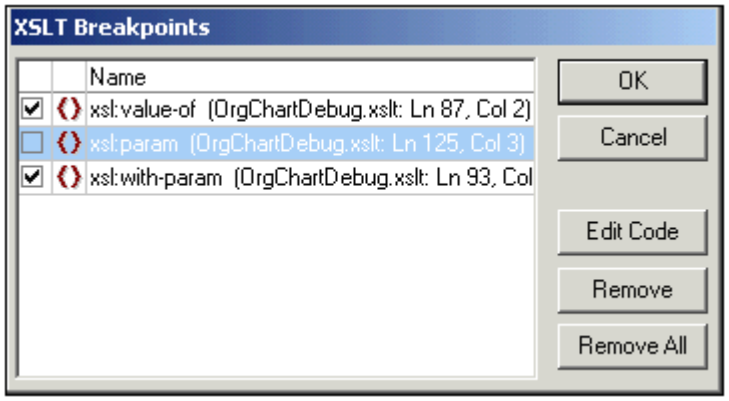
Deaktivieren von Breakpoints:
Sie können Breakpoints, die übersprungen werden sollen, auch deaktivieren ohne den Breakpoint zu löschen.

So deaktivieren Sie einen Breakpoint:

1. Setzen Sie den Cursor in den Node bzw. den Ausdruck, der den Breakpoint enthält.
2. Wählen Sie den Befehl "Breakpoint ein/aus" (Befehl kann auch mit Hilfe der Tastenkombination **Strg+F9** aufgerufen werden). Der Breakpoint ändert sich von rot in grau, was bedeutet, dass er deaktiviert wurde.

Alternativ dazu können Sie Breakpoints auch über das Dialogfeld "Breakpoints" deaktivieren. Gehen Sie dabei vor wie folgt:

1. Wählen Sie den Menübefehl **XSL/XQuery | Breakpoints...**. Daraufhin wird das Dialogfeld "XSLT Breakpoints" geöffnet, in dem die aktuell definierten Breakpoints in allen XML-Quell- und XSLT Stylesheet-Dokumenten angezeigt werden.



2. Entfernen Sie das Häkchen bei den Breakpoints, die Sie deaktivieren möchten und klicken Sie auf OK. Die entsprechenden Breakpoints werden nun statt rot grau angezeigt.

Löschen von Breakpoints
So löschen Sie eine Breakpoint:

1. Setzen Sie den Cursor in den Node oder Ausdruck, der den Breakpoint enthält.
2. Wählen Sie den Befehl "Breakpoint einfügen/löschen" (auch über die Schaltfläche der Symbolleiste oder mit **F9** aufrufbar).

Alternativ dazu können Sie Breakpoints auch über das Dialogfeld "Breakpoints" löschen. Gehen Sie dabei vor wie folgt:

1. Wählen Sie den Menübefehl **XSL | Breakpoints...**.
2. Klicken Sie auf den Breakpoint im Dialogfeld und klicken Sie auf die Schaltfläche "**Löschen**".

Bei Auswahl der Schaltfläche **Alle löschen** werden alle Breakpoints aus dem Dialogfeld (und allen XSLT Stylesheets) gelöscht .

Suchen eines bestimmten Breakpoint
Über das Dialogfeld "Breakpoints" können Sie nach einem bestimmten Breakpoint in einem Dokument suchen. Gehen Sie dabei vor wie folgt:

1. Wählen Sie die Menüoption **XSL/Xquery | Breakpoints...**.
2. Klicken Sie auf den gewünschten Breakpoint in der Liste der Breakpoints.
3. Klicken Sie auf die Schaltfläche **Code bearbeiten**. Das Dialogfeld "Breakpoints" wird geschlossen und der Textcursor wird in der Textansicht direkt vor den Breakpoint gesetzt. In der Enhanced Grid-Ansicht wird die Tabellenzelle mit dem Breakpoint rot markiert.

So setzen Sie den Debugging-Vorgang nach dem Breakpoint fort
Um nach einem Breakpoint mit dem Debuggen fortzufahren, wählen Sie den Befehl **Einsteigen** oder **Debugger starten** (auch über die Symbolleiste aufrufbar).

Hinweis: Breakpoint-Informationen werden in einem Dokument - auch nach dem Schließen der Debugger-Sitzung - so lange beibehalten, bis das Dokument geschlossen wird.

© 2005 Altova GmbH

11 SOAP Debugger

In diesem Abschnitt wird beschrieben wie man:

- SOAP Requests mit Hilfe des SOAP Debuggers sendet und empfängt
- Breakpoints zum Senden und Empfangen von SOAP Requests setzt
- einen fehlerhaften SOAP Request vor dem Absenden an einen Webservice bearbeitet

Beim Entwickeln von Applikationen, die Daten über einen Webservice aufrufen und diese weiterverarbeiten, kann es manchmal zu unerwarteten Problemen kommen. Dabei wäre es natürlich praktisch, wenn man die übertragenen Daten anzeigen und bearbeiten könnte.

Zu diesem Zweck wurde der SOAP Debugger entwickelt. Er fungiert als **Proxy Server** zwischen Client und Webservice. Die Funktionen des Debuggers gehen viel weiter in die Tiefe als die eines normalen Trace-Programms, das nur die Requests und Responses der einzelnen Calls speichert.

Mit Hilfe des SOAP Debuggers können Sie Breakpoints für jede Request und Response Message setzen und mittels XPath-Ausdrücken sogar bedingte Breakpoints setzen.

Der SOAP Debugger unterstützt:

- das schrittweise Überprüfen von SOAP Requests und Responses
- das Bearbeiten der SOAP Requests und Responses und
- das Weiterleiten bearbeiteter Requests an den Client oder Server.

Bitte beachten Sie:
Der SOAP Debugger unterstützt das sichere HTTP-Kommunikationsprotokoll **HTTPS** nicht.

11.1 SOAP-Kommunikationsvorgang

Nach dem Starten des Proxy Servers läuft der SOAP-Kommunikationsvorgang folgendermaßen ab:

- Der Proxy Server überwacht einen Eingang/Port ständig auf eingehende **Client Requests**

1. Die Client-Anwendung **sendet** einen Request an den **Proxy Server**
2. Client Requests können geändert werden, wenn/falls Breakpoints gefunden wurden.
3. Die **Request-Daten** des Proxy Servers werden an den **Webservice-Server** weitergeleitet.

- Der **Websevice Server antwortet** auf den Proxy Server Request und sendet die Response-Daten zurück an den Proxy Server.

1. Die Server Response-Daten können geändert werden, wenn/falls Breakpoints gefunden werden.
2. Die Proxy Server **Response-Daten** werden an die **Client-Applikation** weitergeleitet.

- Die Client Applikation empfängt Response-Daten vom Proxy Server.

Anwendungsbeispiel für Client- und Server-Applikationen:

- Die **Client-**Applikation, die SOAP-Messages sendet und empfängt, ist das Browser-Fenster
- Der "Nanonull Time Service" Dienst, fungiert als **Webservice Server**.
  Die URL für diesen Webservice lautet:
  http://www.nanonull.com/TimeService/TimeService.asmx?WSDL

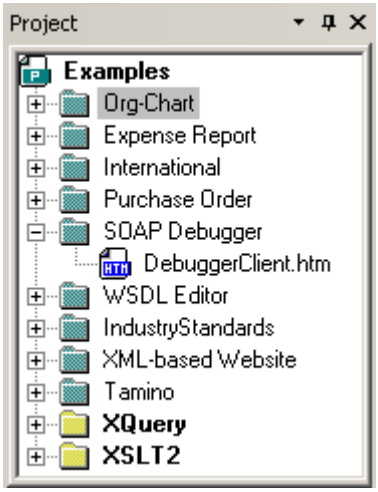
Port-Einstellungen
Der SOAP Debugger verwendet zum Überwachen der Client Request den Port 8080. Der Port kann nur beim Start einer neuen SOAP Debugger-Sitzung geändert werden. Beachten Sie, dass dieser Port durch Firewalls oder Werbefilter deaktiviert sein kann. Sie müssen diese daher zuerst deaktivieren oder einen anderen Port auswählen.

© 2005 Altova GmbH

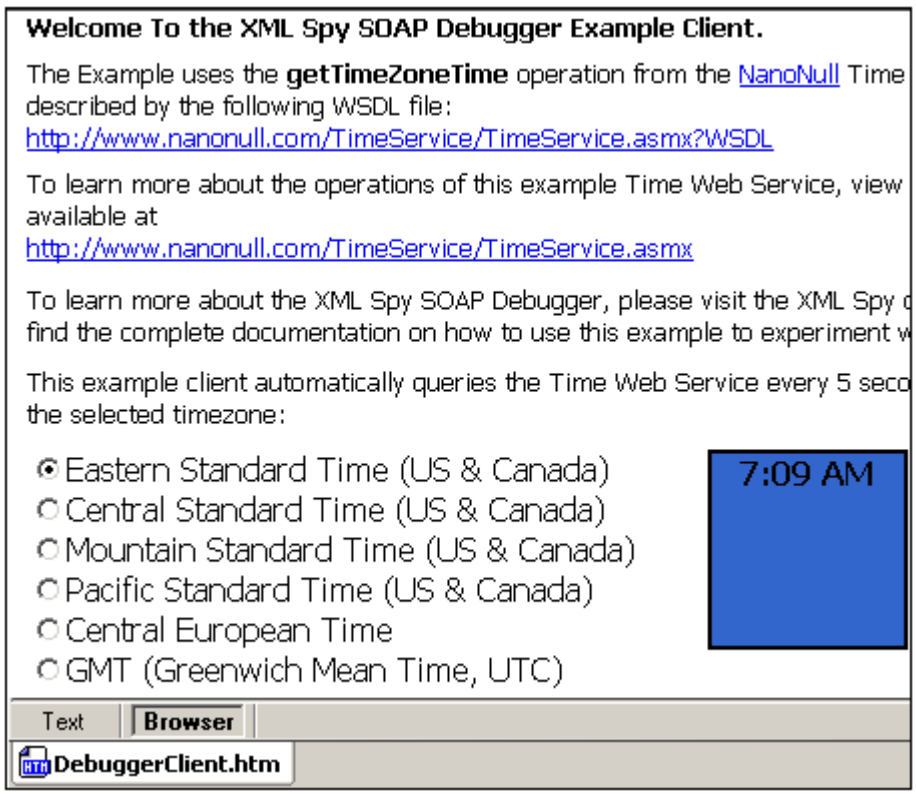
11.2 Verwendung des SOAP Debuggers

Starten Sie XMLSpy 2005 und

1. wählen Sie die Menüoption **Projekt | Öffnen**.
2. Wählen Sie die Datei "**Examples.spp**" im Ordner **...\Examples**.
3. Klicken Sie auf das +-Zeichen des Ordners SOAP Debugger, um den Inhalt des Ordners anzuzeigen.



4. Doppelklicken Sie auf die Datei **DebuggerClient.htm,** um den SOAP Client zu starten und eine Verbindung zum Webservice herzustellen.



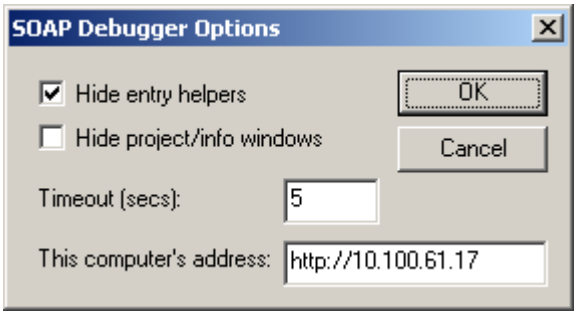
Im **html-**Fenster wird die aktuelle Eastern Standard Time (7:09 in diesem Beispiel) angezeigt.

5. Klicken Sie auf das Optionsfeld GMT, um die aktuelle Greenwich Mean Time (westeuropäische Zeit) anzuzeigen.

An diesem Punkt ist offensichtlich ein Fehler aufgetreten. In der Anzeige der Uhrzeit sollte die korrekte Zeit angezeigt werden. Um die Ursache des Problems zu finden und beheben zu können, müssen wir den SOAP Debugger aktivieren, um die SOAP Messages zu analysieren.

Konfigurieren des SOAP Debuggers

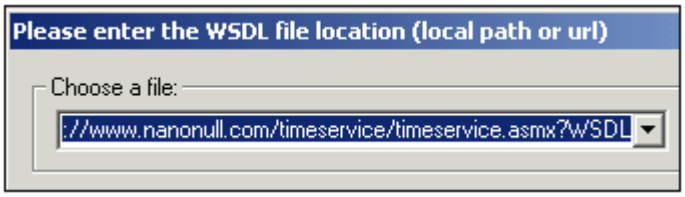
1. Wählen Sie die Menüoption **SOAP | SOAP Debugger-Optionen**.
2. Aktivieren Sie das Kontrollkästchen "Projekt-/Info-Fenster ausblenden" und klicken Sie auf OK.



Auf diese Art haben Sie mehr Platz für das SOAP Request und das SOAP Response-Fenster. Stellen Sie sicher, dass Sie diese Einstellungen vor dem Starten der SOAP Debugger-Sitzung definieren.

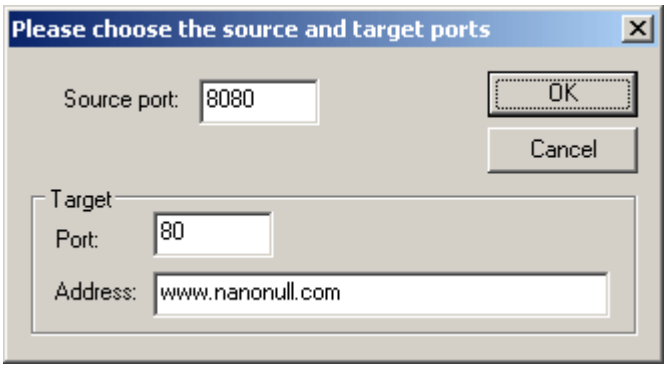
So starten Sie den SOAP Debugger:

1. Wählen Sie die Menüoption **SOAP | SOAP Debugger-Sitzung**.
   Daraufhin wird das Dialogfeld "Enter the WSDL file location..." angezeigt.

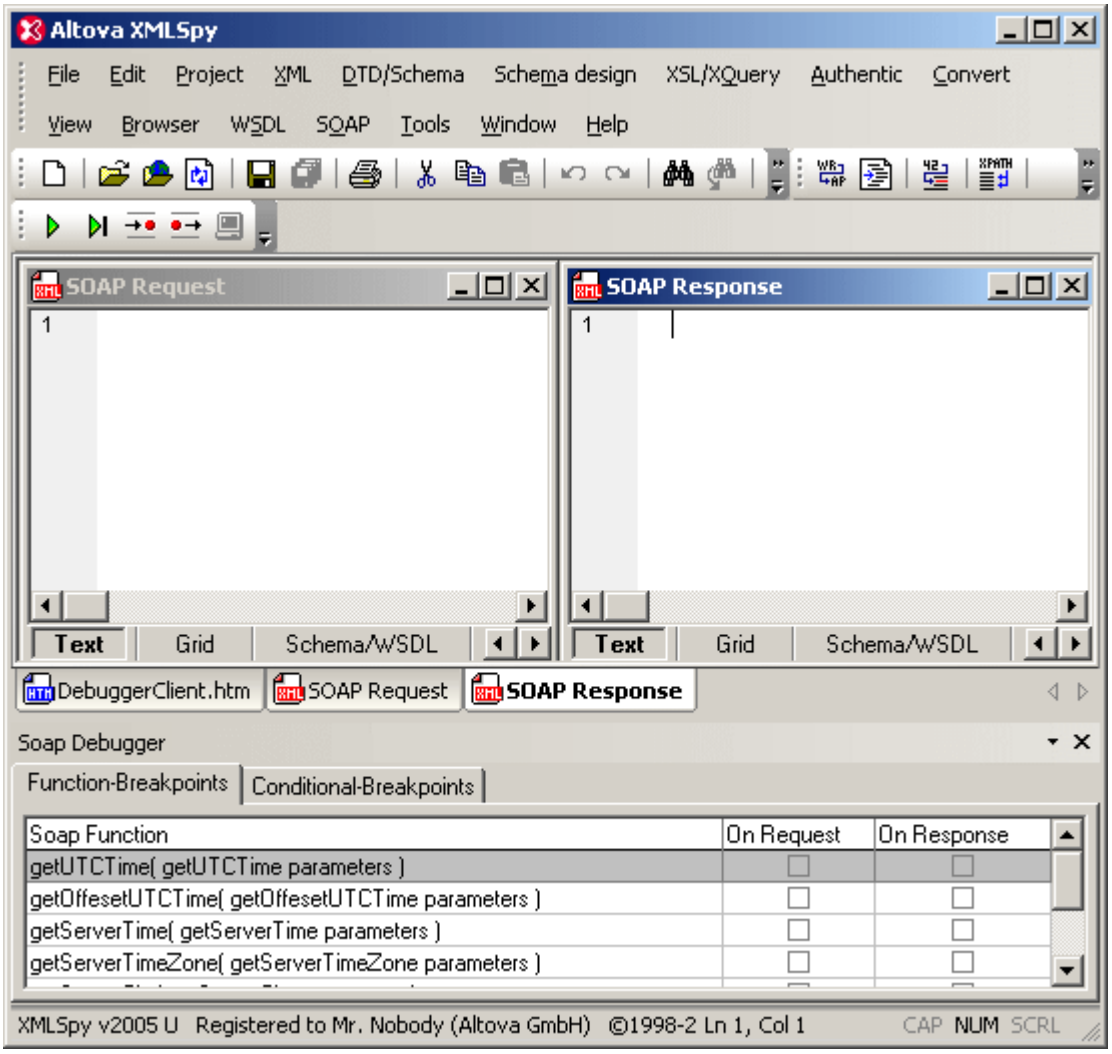


2. Geben Sie den Speicherort der WSDL-Datei ein (wie unten im Textfeld **http://www.nanonull.com/TimeService/TimeService.asmx?WSDL** gezeigt) und klicken Sie auf OK.
   Im folgenden Dialogfeld werden die Einstellungen für den Quell- und Zielport angezeigt. Die **Quell-Port-**Einstellung (8080) ist eine Standardeinstellung und wird automatisch eingegeben. Sie können diese Einstellung jedes Mal, wenn Sie den Debugger starten, ändern.
   Die Einstellung für den **Ziel-Port** stammt aus der WSDL-Datei und wird ebenfalls automatisch ausgefüllt.

© 2005 Altova GmbH



3. Klicken Sie auf die Schaltfläche **OK,** um die SOAP Debugger-Sitzung zu starten.

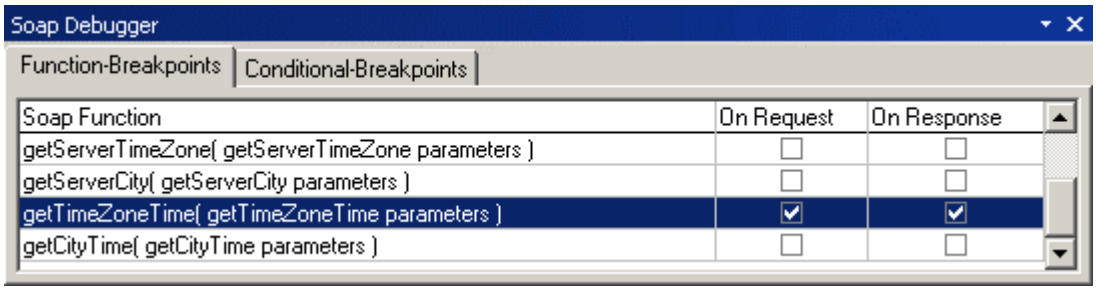


Daraufhin wird der SOAP Debugger Proxy Server im deaktivierten Status geöffnet. Sie sehen den Status daran, dass das Proxy Server-Symbol deaktiviert ist.

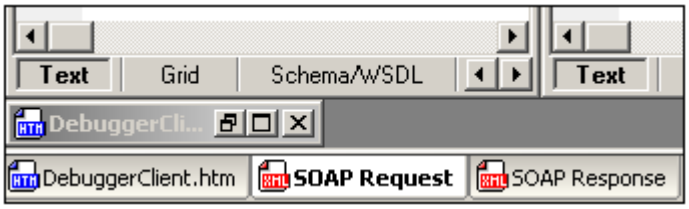
Setzen von Debugger Breakpoints:
Dieser Webservice verwendet die Methode **getTimeZoneTime**, um die Uhrzeit der Zeitzone zu

suchen.

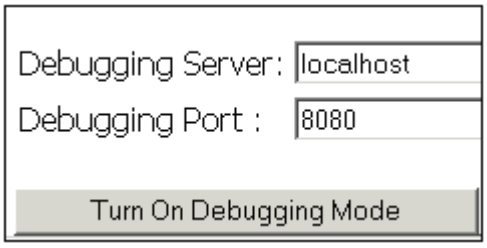
1. Aktivieren Sie die Kontrollkästchen **On Request** und **On Response** für diese Methode (**getTimeZoneTime)**. Auf diese Art können Sie die SOAP Requests und Responses auf Fehler analysieren.



2. Klicken Sie auf das **Start**-Symbol (oder wählen Sie den Menübefehl **SOAP | Starten**).
3. Klicken Sie auf das Register **DebuggerClient** um zum Client zu wechseln. Eventuell müssen Sie **Fenster | DebuggerClient.htm** auswählen, um den Inhalt zu sehen.



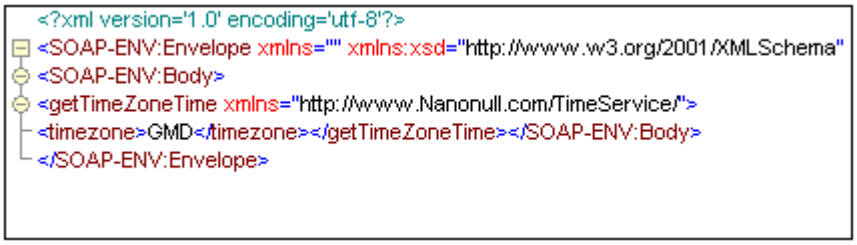
4. Klicken Sie im Client-Fenster auf die Schaltfläche "Turn On Debugging Mode".



Daraufhin wird die Meldung "Debug on" angezeigt und der SOAP Request wird an den Debugger gesendet.

Anzeigen und Editieren des SOAP Request
Der SOAP Request wird automatisch im SOAP Request-Fenster des Debuggers in der Textansicht angezeigt. Wir wollen uns den Request nun ansehen und etwaige Fehler beheben.

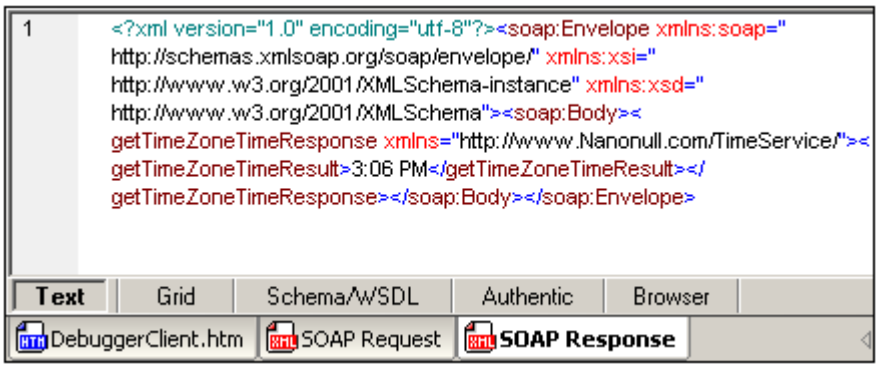
 © 2005 Altova GmbH



Bei Element **timezone** stellen wir fest, dass der Wert **GMD** ist. Dies ist falsch und muss in GMT geändert werden.

1. Doppelklicken Sie in das Feld **timezone**, ändern Sie den Eintrag in **GMT** und drücken Sie die Eingabetaste.

2. Klicken Sie auf das **Start-**Symbol (oder wählen Sie den Menübefehl **SOAP | Starten**), um den **korrigierten Request** an den Webservice zu senden.
Nach wenigen Sekunden erscheint der **Webservice Response** auf den SOAP Request im SOAP Response-Fenster. Wählen Sie **Ansicht | Zeilenumbruch**, um den Inhalt zu sehen (siehe Diagramm unten).



Wir haben im Response keine Fehlermeldung erhalten, daher ist anzunehmen, dass es sich um die richtige Uhrzeit handelt.

3. Klicken Sie auf das Debugger **Start**-Symbol , um diesen Response an den **Client** zu senden (in diesem Fall das Browser-Fenster).
4. Klicken Sie auf das Register **DebuggerClient.htm**, um wieder zum Client zu wechseln und überprüfen Sie das neue Resultat.

Die Fehlermeldung wird nicht mehr angezeigt, stattdessen sehen Sie die korrekte GMT-Zeit.

5. Wählen Sie den Menübefehl **SOAP | SOAP Debugger-Sitzung**, um die Debugger-Sitzung zu beenden.

Fehlerbehebung

Da wir nun wissen, dass der **Client** den unzulässigen Wert "GMD" anstatt "GMT" in seinem Request sendet und dass der Server auf "GMT" den korrekten Response senden würde, wissen wir, dass das Problem beim SOAP Client liegt und wir können das Problem nun beheben.

Gehen Sie im Fenster **DebuggerClient.html** folgendermaßen vor:

1. Klicken Sie auf die Schaltfläche "Text", öffnen Sie das Dialogfeld "Suchen" (**Bearbeiten | Suchen**) und suchen Sie nach "GMD".
   In XMLSpy 2005 wird das folgende Code-Fragment angezeigt:

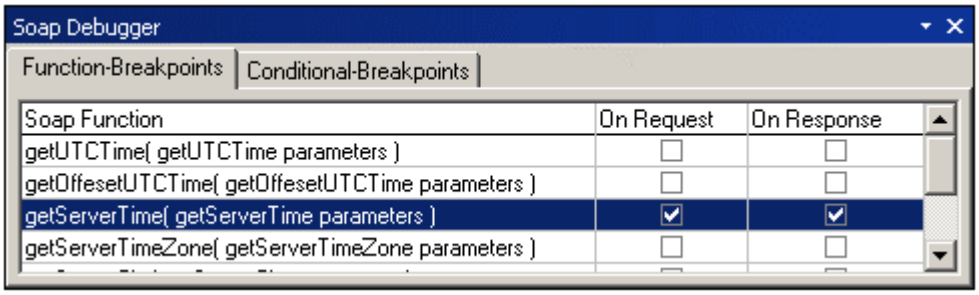
```
138         else if(timezone[4].checked)
139             msCurrentTimeZone='CET';
140         else if(timezone[5].checked)
141             msCurrentTimeZone='GMD';
142         }
```

2. Ändern Sie den Wert von "GMD" in "GMT" und wechseln Sie wieder in die Browser-Ansicht.
   Der SOAP Client arbeitet nun ordnungsgemäß, die Meldung "Unknown Time zone" wird nicht mehr angezeigt. Stattdessen erscheint im Fenster die richtige Uhrzeit. Somit ist das Problem behoben.

© 2005 Altova GmbH

11.3 Setzen von Breakpoints im SOAP Debugger

Das Setzen und Löschen von Breakpoints erfolgt im SOAP Debugger-Fenster. Es besteht aus den folgenden zwei Registerkarten:



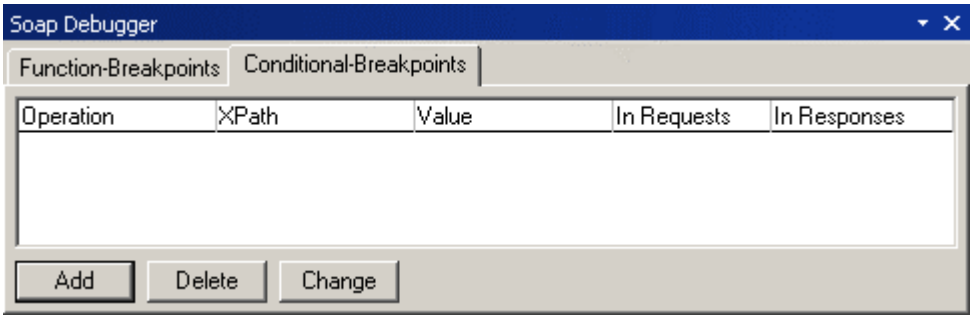
Functional-Breakpoints
Das Register "Functional breakpoints" gestattet Ihnen, in Requests und/oder Responses auf SOAP-Methoden Breakpoints zu setzen. Der Debugger markiert die Funktionszeile, die den Breakpoint ausgelöst hat.

Datenpakete, die vom und an den Client gesendet werden, werden analysiert und den entsprechenden Funktionen aus der WSDL-Datei zugeordnet. Wenn ein Breakpoint für eine bestimmte Methode gesetzt wird, dann hält der SOAP Debugger an dieser Stelle an. Die Symbolleistenschaltflächen werden zu diesem Zeitpunkt aktiviert.

Die Daten werden in den Dokumentenfenstern "Soap Request" oder "Soap Response" angezeigt. Die in den SOAP-Fenstern angezeigten SOAP-Dokumente können nun geändert werden. Die Daten werden gesendet, sobald Sie auf eine der Symbolleistenschaltflächen (mit Ausnahme des Symbols "Proxyserver anhalten") klicken.

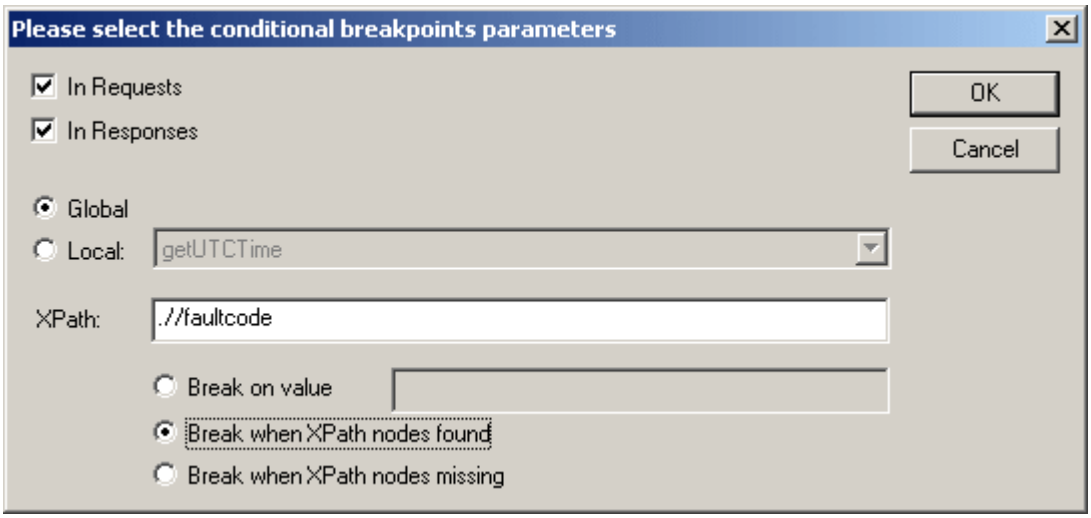
Conditional-Breakpoints
Auf dem Register "Conditional Breakpoints" können Sie XPath-Ausdrück verwenden, um Breakpoints zu definieren.

Wenn ein SOAP **Request** einen Fehler verursacht, muss der SOAP **Response** ein "faultcode"-Element enthalten. Wir möchten daher, dass an jedem faultcode-Element ein Breakpoint ausgelöst wird.



So fügen Sie einen bedingten Breakpoint (conditional Breakpoint) hinzu:

1. Klicken Sie auf das Register "Conditional Breakpoints" und anschließend auf die Schaltfläche **Add.**



2. Geben Sie den XPath-Ausdruck (z.B. **.//faultcode**) in das Feld XPath ein.
3. Klicken Sie auf das Optionsfeld "Anhalten wenn XPath Node gefunden".
4. Bestätigen Sie die Einstellungen durch Klicken auf OK.
   Der SOAP Debugger hält nun immer dann an, wenn im SOAP Request oder Response ein **.//faultcode-**Element erscheint.

Eintreffende Requests
Mit Hilfe dieser Option können Sie die jeweilige Aktion definieren, die bei Auftreten eines Breakpoint in einem SOAP Request durchgeführt werden soll. Der Rest der Optionen des Dialogfelds bezieht sich auf den SOAP Request. Natürlich können Sie dieselben Einstellungen auch für SOAP Responses verwenden, wenn Sie auch das Kontrollkästchen "Eintreffende Responses" aktivieren.

Eintreffende Responses
Mit Hilfe dieser Option können Sie die jeweilige Aktion definieren, die bei Auftreten eines Breakpoint in einem SOAP Response durchgeführt werden soll. Der Rest der Optionen des Dialogfeld bezieht sich auf den SOAP Response. Natürlich können Sie dieselben Einstellungen auch für SOAP Requests verwenden, wenn Sie auch das Kontrollkästchen "Eintreffende Requests" aktivieren.

Global
Bei Auswahl der Option "Global" wird jede Methode/Funktion auf die von Ihnen definierte Bedingung durchsucht.

Lokal
Bei Auswahl der Option "Lokal" müssen Sie eine bestimmte Methode/Funktion aus den Methoden/Funktionen der WSDL-Datei auswählen. Diese Methode/Funktion wird dann nach der von Ihnen definierten Bedingung durchsucht.

XPath
Geben Sie hier den gewünschten XPath-Ausdruck/Node ein. Sie müssen hier einen XPath eingeben, um eine der Optionen der Optionsfelder verwenden zu können.

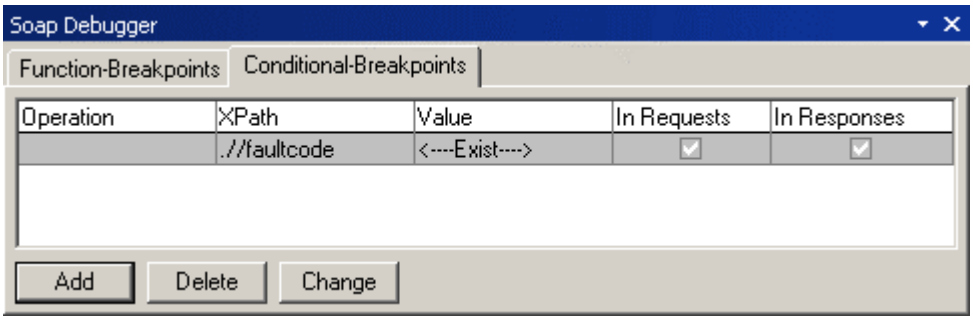
Anhalten bei Wert
Der Debugger hält an, wenn einer der XPath Nodes mit dem in diesem Feld angegebenen Wert übereinstimmt.

Anhalten wenn XPath Node gefunden
Der Debugger hält an, wenn der XPath Node **.//faultcode**, der im XPath-Feld eingegeben

 © 2005 Altova GmbH

wurde, im SOAP Request oder dem SOAP Response **vorhanden** ist.

Anhalten bei fehlenden XPath Nodes
Der Debugger hält an, wenn der XPath Node **.//faultcode**, der im XPath-Feld eingegeben wurde, im SOAP Request oder dem SOAP Response **nicht vorhanden** ist.



Operation
Die Spalte "Operation" enthält die Methode, nach der gesucht wird. Wenn Sie das Optionsfeld **Global** aktiviert haben, bleibt dieses Feld leer. Bei Auswahl von **Lokal** wird die gewählte Methode hier angezeigt.

XPath
Diese Spalte enthält den von Ihnen definierten XPath-Ausdruck.

Value
Diese Spalte enthält den XPath-Wert, auf den die XPath Nodes überprüft werden. Wenn Sie die Option "Anhalten bei Wert" gewählt haben, wird der von Ihnen eingegebene String hier angezeigt. Wenn Sie die Option Anhalten wenn XPath Node gefunden" gewählt haben, so wird <--Exist--> angezeigt. Wenn Sie die Option "Anhalten bei fehlenden XPath Nodes" gewählt haben, wird <--Missing--> angezeigt.

In Requests
Das Kontrollkästchen wird angezeigt, wenn der XPath-Ausdruck im SOAP Request mit einem Häkchen versehen ist.
Sie können die Einstellungen ändern, indem Sie das Kontrollkästchen direkt in der Spalte aktivieren.

In Responses
Das Kontrollkästchen wird angezeigt, wenn der XPath-Ausdruck im SOAP Response mit einem Häkchen versehen ist.
Sie können die Einstellungen ändern, indem Sie das Kontrollkästchen direkt in der Spalte aktivieren.

So bearbeiten Sie die bedingten Breakpoints:

- Doppelklicken Sie auf die entsprechende Zeile im Register oder
- Klicken Sie auf die Schaltfläche "Change".

So löschen Sie einen bedingten Breakpoint

- Klicken Sie auf die zu löschende Zeile und anschließend auf die Schaltfläche "Delete".

12 Code Generator

XMLSpy 2005 enthält nun einen integrierten Code Generator, der aus XML-Schemadefinitionen, automatisch Java-, C++ oder C#-Klassendateien generieren kann.

XML ist keine Programmiersprache im eigentlichen Sinne, da es nicht kompiliert oder als eigenständige ausführbare Binärdatei ausgeführt werden kann; XML-Dokumente müssen in eine externe Software-Applikation oder Runtime-Umgebung wie z.B. eine Business-to-Business-Applikation oder einen Web Service eingebunden werden.

Für die Implementierung jeder benutzerdefinierten XML-Software-Applikation müssen letztendlich innerhalb Ihres Codes Programmzugriffsmethoden implementiert werden, um speicherresidente Operationen an einem XML-Dokument erstellen, validieren, verarbeiten, transformieren oder durchführen zu können.

Das Schreiben von Programmiercode erfolgt normalerweise über APIs (Application Program Interfaces) zur Verarbeitung von XML auf höchster Ebene, wie z.B. Microsoft MSXML und Apache Xerces Version 2.2.0, die für verschiedene Programmiersprachen kostenlos zur Verfügung stehen.

© 2005 Altova GmbH

12.1 Einführung in Code Generator

Im Fall von XML-Schemas generiert die Standardvorlage von XMLSpy 2005 Code Generator automatisch Klassendefinitionen für alle deklarierten Elemente oder complexTypes. Diese Definitionen definieren alle complexTypes in Ihrem Schema neu, wobei die durch die Erweiterungen der complexTypes in Ihrem XML-Schema definierten Klassenableitungen erhalten bleiben.

Bei komplexen Schemas, die Schemakomponenten aus mehreren Namespaces importieren, behält XMLSpy 2005 diese Informationen durch Generierung der entsprechenden C#-, C++-Namespaces oder Java-Pakete bei.

Zusätzlicher Code wird implementiert, wie z.B. Funktionen, die XML-Dateien in eine speicherresidente DOM (Document Object Model)-Repräsentation auslesen, XML-Dateien von einer DOM-Repräsentation zurück in eine Systemdatei schreiben, sowie XML-Validierung und Transformation.

Der erzeugte Programmcode wird in den Programmiersprachen C++, Java oder C# ausgegeben. Der generierte C++-Ausgabecode verwendet entweder MSXML 4.0 oder Xerces und enthält eine Visual Studio 6.0-Projektdatei; der erzeugte Java-Code wird für den Industriestandard JAXP (Java API for XML Parsing) geschrieben und enthält eine Sun Forte-Projektdatei für Java, eine JBuilder-Projektdatei und eine ANT-Konfigurationsdatei.

Bei der C++-Codegenerierung wird sowohl in MapForce 2005 als auch in XMLSpy 2005 Visual C++ 7.1 / Visual Studio .NET 2003 direkt unterstützt.
Zusätzlich zu den neuen **.dsw /.dsp-**Dateien für Visual Studio 6.0 werden auch immer eine **.sln** und .**vcproj**-Datei generiert.

Der C#-Code bedient sich der .NET XML-Klassen und kann von jeder .NET-fähigen Programmiersprache aus verwendet werden (z.B. VB.NET, Managed C++, J# oder jeder beliebigen der verschiedenen für die .NET-Plattform verfügbaren Sprachen).

Resultat in XMLSpy 2005:

Resultat	**Speicherort**	**XML Spy 2005**
Standardbibliotheken	Ordner "Altova"	þ
Schema Wrapper-Bibliotheken	Schema-Name-Ordner	þ
Applikation		
Test Applikation - Quellcode mit Beispielen	Die Testapplikation muss editiert werden, damit der Code funktioniert. Zu diesem Zeitpunkt nicht funktionsfähig.	þ

Code Generator-Vorlagen
Ausgabecode kann über eine einfache aber leistungsfähige Template-Sprache vollständig vom Benutzer angepasst werden. Diese Template-Sprache bietet volle Kontrolle über das Mappen von in XML-Schemas integrierten Datentypen auf die primitiven Datentypen einer bestimmten Programmiersprache.
Sie können damit den zugrunde liegenden Parse- und Validierungsprozessor problemlos ersetzen, Code entsprechend den Codekonventionen Ihres Unternehmens anpassen oder verschiedene Basisbibliotheken wie z.B. die Microsoft Foundation Classes (MFC) und die Standard Template Library (STL) verwenden.

Zusätzlich dazu können Sie Ihre eigenen Vorlagen erzeugen, um beliebigen Code wie EJB's, WSDL-Dateien, SQL Scripts, ASP oder WML Code generieren zu können.

Der Schema Editor von XMLSpy 2005 eignet sich gut als Tool zum Entwerfen eines Software-Modells oder Prototyps, da er es gestattet, auf einer hohen Ebene im XML-Schema rasch einen Prototyp einer XML-Applikation zu erzeugen und XML-Applikationen automatisch zu generieren. Änderungen am Inhaltsmodell eines XML-Schemas einer Applikation können einfach durch erneutes Ausführen des Code Generator sofort mit einer Software-Implementierung in Einklang gebracht werden.

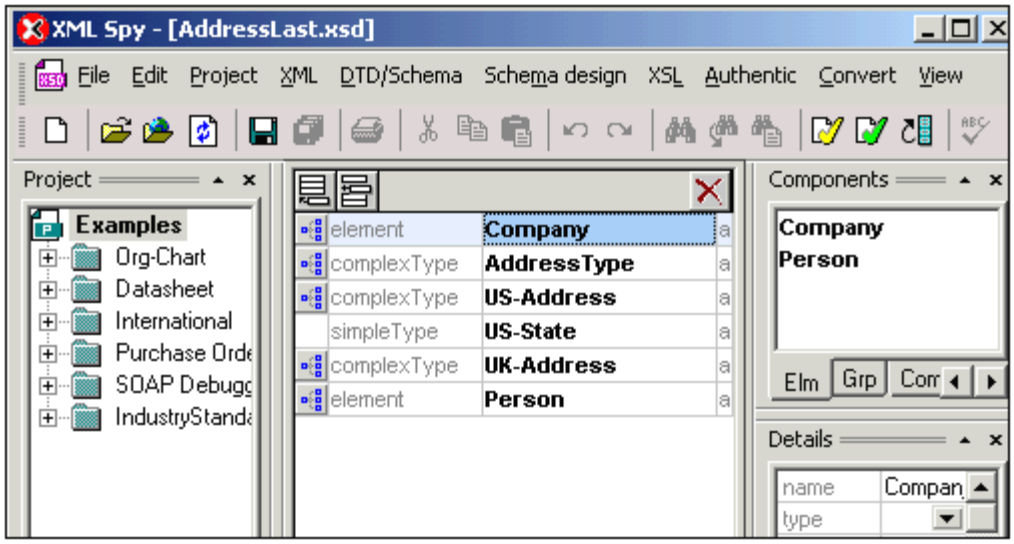
Dank der integrierten Code Generierung müssen sich Software Developer nicht mehr um das Schreiben von Infrastruktur-Code auf niedriger Ebene kümmern, sondern können sich ganz der Implementierung geschäftskritischer Software widmen. Durch automatische Generierung eines Programming Language Binding beschleunigt XMLSpy 2005 die Projektentwicklung zwischen Anfangsentwurf und endgültiger Implementierung, wodurch erhebliche Kosten- und Zeitersparnisse und somit ein Marktvorteil erzielt werden können.

© 2005 Altova GmbH

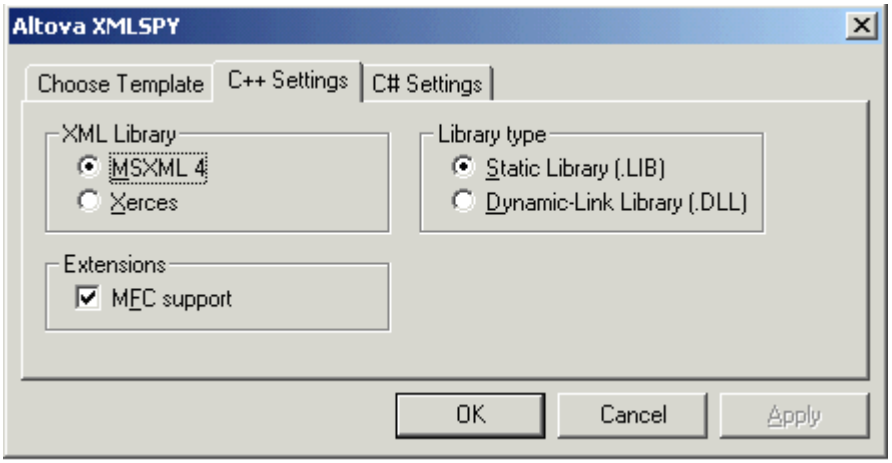
12.2 Generieren von Quellcode anhand eines Schemas

In der folgenden Anleitung wird beschrieben, wie man anhand der Schemadatei AddressLast als Input-Datei Quellcode generiert und kompiliert.

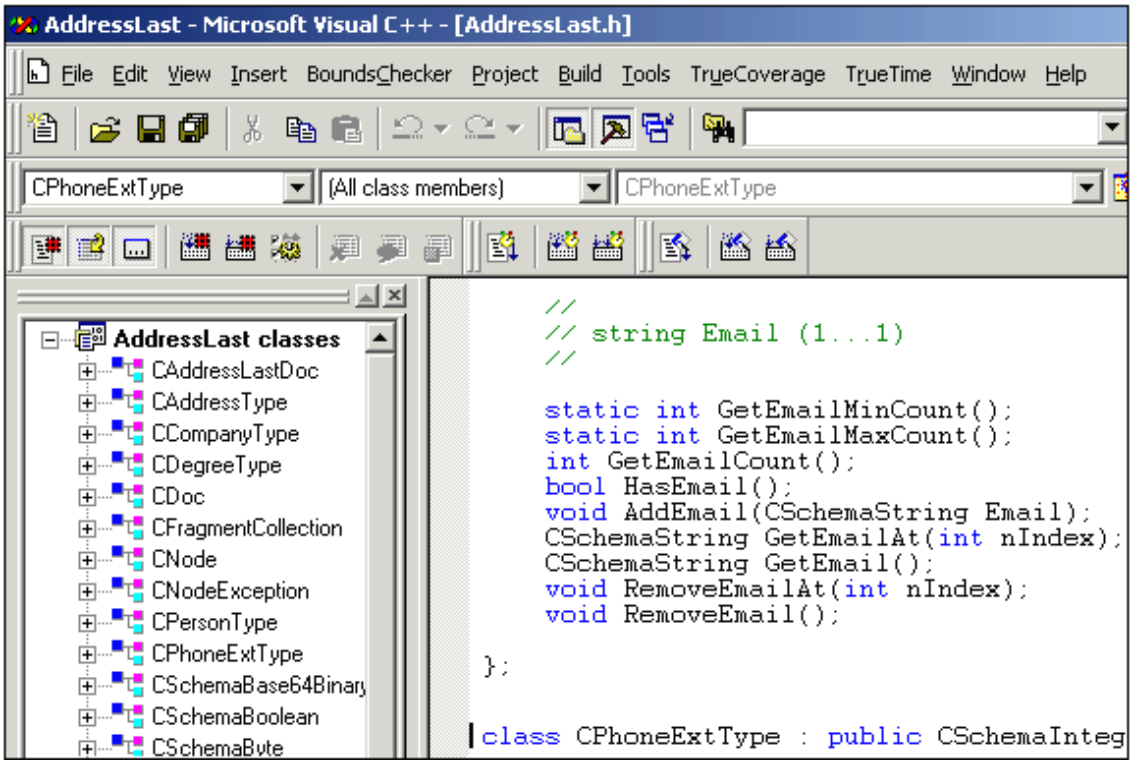
1. Öffnen Sie das Schema, für das Sie Quellcode generieren möchten (z.B. AddressLast).



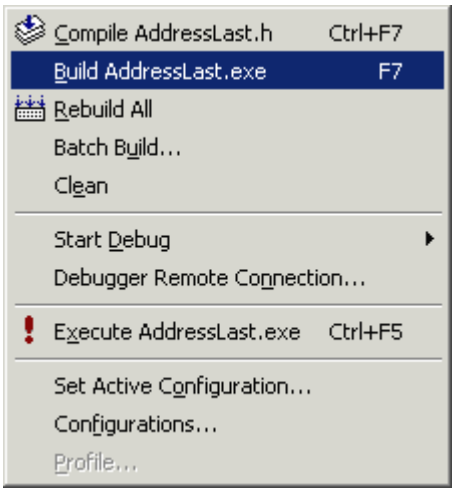
2. Wählen Sie den Menüpunkt **DTD/Schema | Programmcode generieren...**.
3. Wählen Sie den zu generierenden Code aus: Java, C++ oder C#, (z.B. java.spl, cpp.spl oder cs.spl). Auf der Registerkarte C++ Einstellungen können Sie die Einstellungen für die Generierung von C++-Code näher definieren.



4. Wählen Sie im Dialogfeld "Speichern als" den Ordner und Dateinamen (AddressLast.dsw) aus und klicken Sie auf Speichern.
   Wenn Sie Java-Code erzeugen, wird lediglich das Dialogfeld zur Auswahl des Ordners angezeigt.
5. Sie werden nun aufgefordert, das neu erstellte Projekt in Microsoft Visual Studio zu öffnen. Klicken Sie auf OK.
   Bei Erzeugung von Java-Code werden Sie aufgefordert, das entsprechende Ausgabeverzeichnis zu öffnen.



6. Wählen Sie die Menüoption **Erzeugen | Erzeuge AddressLast.exe**.



7. Der Code wird kompiliert und verknüpft.

 © 2005 Altova GmbH

```
--------------------Configuration: AddressLast - Win32 Unicode Release-
Compiling resources...
Compiling...
StdAfx.cpp
Compiling...
AddressLast.cpp
AddressLastTest.cpp
Node.cpp
Generating Code...
Linking...
LINK : warning LNK4089: all references to "OLEAUT32.dll" discarded by

AddressLast.exe - 0 error(s), 1 warning(s)
```

8. Drücken Sie die Taste **F5** oder klicken Sie auf das Symbol **Starten,** um die Datei zu starten.
9. Eine Meldung über mangelnde Debugging Informationen wird angezeigt. Klicken Sie auf OK, um fortzufahren.

   Die Datei AddressLast.exe wird erfolgreich kompiliert.

```
Loaded 'C:\WINNT\system32\msvcrt.dll', no matching symbolic information found.
Loaded 'C:\WINNT\system32\MSVCP60.DLL', no matching symbolic information found
Loaded 'C:\WINNT\system32\imm32.dll', no matching symbolic information found.
Loaded 'C:\WINNT\system32\Bccore.dll', no matching symbolic information found.
The thread 0x384 has exited with code 0 (0x0).
The program 'C:\Program Files\Altova\xmlspy\CodeGen\UnicodeRelease\AddressLast'
```

12.3 Beispiele für die Verwendung des generierten Java-, C++ und C#-Codes

Verwendung der generierten Klassen
Im vorherigen Abschnitt wurde beschrieben, wie man ein Schema öffnet, eine Vorlage auswählt und die benötigten Dateien generiert. Anschließend öffnen Sie das generierte Projekt bzw. das entsprechende Verzeichnis und können mit der Arbeit beginnen.

Der Einfachheit halber werden wir mit einer sehr kleinen XSD-Datei arbeiten. Unten sehen Sie den Quellcode dazu. Speichern Sie die Datei als "**Library.xsd**", wenn Sie dieselben Ergebnisse wie in diesem Beispiel erhalten wollen:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.nanonull.com/LibrarySample"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.nanonull.com/LibrarySample" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Library">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Book" type="BookType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="BookType">
        <xs:sequence>
            <xs:element name="Title" type="xs:string"/>
            <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="ISBN" type="xs:string" use="required"/>
    </xs:complexType>
</xs:schema>
```

In diesem Beispiel werden wir nur diese Schema-Datei verwenden, damit man die Unterschiede zwischen den verschiedenen Programmiersprachen in den folgenden Beispielen leicht erkennt.

Beachten Sie bitte, dass die generierten Klassen von XSD-Datei zu XSD-Datei unterschiedlich benannt werden. Stellen Sie sicher, dass Sie die Namen der Funktionen und Variablen anpassen, wenn Sie mit Ihren eigenen Schemas oder anderen Beispielen arbeiten.

12.3.1 Erzeugen von XML-Dateien

Java:

Um Ihnen eine Übersicht zu geben, finden Sie unten den Code für die Erzeugung der Testdatei:

```
protected static void example() {
        LibraryType lib = new LibraryType();
        BookType book = new BookType();
        book.addISBN( new SchemaString( "0764549642" ) );
        book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
        book.addAuthor( new SchemaString( "Larry Kim" ) );
        lib.addBook( book );
```


© 2005 Altova GmbH

```
        LibraryDoc doc = new LibraryDoc();
        doc.setRootElementName( "http://www.nanonull.com/LibrarySample", "Library"
);
        doc.setSchemaLocation( "Library.xsd" ); // optional
        doc.save( "Library1.xml", lib );
}
```

Den generierten Klassen liegt die folgende Idee zugrunde:
Generieren Sie die Baumstruktur, in der die neue Datei gespeichert werden soll, erstellen Sie ein Dokument und speichern Sie die Baumstruktur darin.

Die erste Zeile:

```
LibraryType lib = new LibraryType();
```

LibraryType ist eine Klasse, die generiert wurde. Sie ist dem Element-Namen <xs:element name="Library"> in der Schema-Datei ähnlich. Dies ist das Root-Element der generierten XML-Dateien (wie beim Generieren einer XML-Beispieldatei).

Wir haben nun ein neues Library-Objekt. Nun müssen wir dieses nur noch mit Book-Objekten befüllen. Wenn Sie sich das Schema noch einmal ansehen, werden Sie feststellen, dass "Books" als eine Sequenz innerhalb der Library gespeichert werden.

Der nächste Schritt ist, ein Objekt von **BookType** zu erstellen und es auszufüllen.

```
BookType book = new BookType();
```

Definition von BookType in der Schema-Datei:

```
<xs:complexType name="BookType">
    <xs:sequence>
        <xs:element name="Title" type="xs:string"/>
        <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ISBN" type="xs:string" use="required"/>
</xs:complexType>
```

Wir fügen nun eine ISBN-Nummer, einen Titel und einen Autor ein.

```
book.addISBN( new SchemaString( "0764549642" ) );
book.addTitle( new SchemaString( "The XML Spy Handbook" ) );
book.addAuthor( new SchemaString( "Larry Kim" ) );
```

Wir haben das Objekt "BookType" nun befüllt und müssen es nun zur Sequenz von **internationalPriceTypes** in Test_Empty hinzufügen:

```
lib.addBook(book);
```

Fertig, die Baumstruktur ist erstellt! Nun müssen wir nur noch ein neues Dokument erstellen und den Namespace und Namen des Root-Elements definieren:

```
        LibraryDoc doc = new LibraryDoc();
        doc.setRootElementName( "http://www.nanonull.com/LibrarySample", "Library"
);
```

Optional können wir auch den Speicherort des Schemas festlegen.

```
doc.setSchemaLocation( "Library.xsd" );
```

Speichern Sie die Datei nun. Der erste Parameter ist der Dateiname, der zweite ist das Root-Element selbst.

```
doc.save( "Library1.xml", lib );
```

C++:

Die C++-Implementierung ist der Java-Implementierung sehr ähnlich. Daher werden wir nur auf die Unterschiede zwischen den beiden näher eingehen.

Die oberhalb der Hauptfunktion einzufügende Funktion lautet wie folgt:

```
void Example()
{
	CLibraryType lib;
	CBookType book;
	book.AddISBN( _T("0764549642") );
	book.AddTitle( _T("The XML Spy Handbook") );
	book.AddAuthor( _T("Larry Kim") );
	lib.AddBook( book );

	CLibraryDoc doc;
	doc.SetRootElementName( _T("http://www.nanonull.com/LibrarySample"),
_T("Library") );
	doc.SetSchemaLocation( _T("Library.xsd") ); // optional
	doc.Save( _T("Library1.xml"), lib );
}
```

Als erstes fällt auf, dass alle Klassen mit einem großgeschriebenen C beginnen. Der Rest der Klassennamen bleibt gleich. Alle Member Function-Namen beginnen mit einem Großbuchstaben.

Seien Sie bitte vorsichtig mit den **_T-**Makros. Das Makro stellt sicher, dass die String-Konstante korrekt gespeichert wird, ob Sie nun für Unicode oder Nicht-Unicode-Programme kompilieren.

C#:

```
protected static void Example()
{
	LibraryType lib = new LibraryType();
	BookType book = new BookType();
	book.AddISBN( new SchemaString( "0764549642" ) );
	book.AddTitle( new SchemaString( "The XML Spy Handbook" ) );
	book.AddAuthor( new SchemaString( "Larry Kim" ) );
	lib.AddBook( book );

	LibraryDoc doc = new LibraryDoc();
	doc.SetRootElementName( "http://www.nanonull.com/LibrarySample", "Library"
);
	doc.SetSchemaLocation( "Library.xsd" ); // optional
	doc.Save( "Library1.xml", lib );
}
```


© 2005 Altova GmbH

12.3.2 Öffnen und Parsen vorhandener XML-Dateien

Wir werden nun mit der mit unserem Beispielprogramm erstellten Datei **Library1.xml** arbeiten. Stellen Sie sicher, dass diese Datei vorhanden ist oder ändern Sie im load-Befehl den Pfad zur Datei.

Java:

Beispiel-Quellcode:

```
protected static void example2() {
        LibraryDoc doc = new LibraryDoc();
        LibraryType lib = new LibraryType( doc.load( "Library1.xml" ) );

        if( !lib.hasBook() ) {
                System.out.println( "This library is empty" );
                return;
        }

        for( int i = 0; i < lib.getBookCount(); i++ ) {
                BookType book = lib.getBookAt( i );

                System.out.println( "ISBN:   " + book.getISBN().asString() );
                System.out.println( "Title:  " + book.getTitle().asString() );

                for( int j = 0; j < book.getAuthorCount(); j++ ) {
                        System.out.println( "Author: " + book.getAuthorAt( j ).asString()
);
                }
        }
}
```

Fügen Sie in der ***main()*** function einen Call zu:

```
example2();
```

hinzu nach

```
example1();
```

Der Rest ist relativ einfach.
Erstellen Sie zuerst eine neue Variable vom Typ Document sowie eine für das Root-Element, das sofort durch die Funktion **doc.load** befüllt wird. Der einzige dazu benötigte Parameter ist der Pfad zur verwendeten XML-Datei.

```
LibraryDoc doc = new LibraryDoc();
LibraryType lib = new LibraryType( doc.load( "Library1.xml" ) );
```

Wir haben nun ein Dokument und wissen, dass "lib" eine Sequenz von BookType-Objekten enthält.

Der nächste Schritt ist, die Werte aller Book-Elemente auszugeben:

```
if( !lib.hasBook() ) {
```

Warum soll auf Objekte überprüft werden? Der Call zeigt nur, dass eine Funktion existiert, die dies für Sie überprüft.

Iterieren Sie durch die Books und geben Sie deren Werte aus:

```
        for( int i = 0; i < lib.getBookCount(); i++ ) {
                BookType book = lib.getBookAt( i );

                System.out.println( "ISBN:   " + book.getISBN().asString() );
                System.out.println( "Title:  " + book.getTitle().asString() );

                for( int j = 0; j < book.getAuthorCount(); j++ ) {
                        System.out.println( "Author: " + book.getAuthorAt( j ).asString()
);
                }
        }
```

Wir sind nun fertig. Alle Werte werden in Ihrem Ausgabefenster aufgelistet.

C++:

In diesem Abschnitt werden nur die Unterschiede zum Java-Code erläutert.

```
void Example2()
{
        CLibraryDoc doc;
        CLibraryType lib = doc.Load( _T("Library1.xml") );

        if( !lib.HasBook() )
        {
                cout << "This library is empty" << endl;
                return;
        }

        for( int i = 0; i < lib.GetBookCount(); i++ )
        {
                CBookType book = lib.GetBookAt( i );

                cout << "ISBN:   " << book.GetISBN() << endl;
                cout << "Title:  " << book.GetTitle() << endl;

                for( int j = 0; j < book.GetAuthorCount(); j++ )
                {
                        cout << "Author: " << book.GetAuthorAt( j ) << endl;
                }
        }
}
```

Alle Member Functions beginnen mit einem Großbuchstaben und alle Klassennamen beginnen mit einem C.

C#:

In diesem Abschnitt werden nur die Unterschiede zum Java-Code erläutert.

```
protected static void Example2()
{
        LibraryDoc doc = new LibraryDoc();
        LibraryType lib = new LibraryType( doc.Load( "Library1.xml" ) );

        if( !lib.HasBook() )
        {
```


© 2005 Altova GmbH

```
                Console.WriteLine( "This library is empty" );
                return;
        }

        for( int i = 0; i < lib.GetBookCount(); i++ )
        {
                BookType book = lib.GetBookAt( i );

                Console.WriteLine( "ISBN:   {0}", book.GetISBN() );
                Console.WriteLine( "Title:  {0}", book.GetTitle() );

                for( int j = 0; j < book.GetAuthorCount(); j++ )
                {
                        Console.WriteLine( "Author: {0}", book.GetAuthorAt( j ) );
                }
        }
}
```

Alle Member Functions beginnen mit einem Großbuchstaben.

12.4 Der Weg zu SPL (Spy Programming Language)

In diesem Abschnitt erhalten Sie eine Übersicht über Spy Programming Language (SPL).

Es wird vorausgesetzt, dass Sie bereits über Programmiererfahrung verfügen und mit Operatoren, Funktionen, Variablen und Klassen sowie den Grundzügen von in SPl häufig verwendeter objektorientierter Programmierung vertraut sind. Außerdem sollten Sie gute XML-Schema-Kenntnisse haben.

Im Ordner **..\xmlspy\spl** finden Sie ein Beispiel C++ (und Java-) **cpp.spl.** Anhand dieser Datei können Sie Ihre eigenen Vorlagen entwickeln.

Funktionsweise von Code Generator:
Die Basis von Code Generator bilden die Vorlagendateien (.spl). Die Vorlagendatei wird von Code Generator interpretiert und je nach Vorlage wird eine **.cpp, .java** Quellcodedatei (oder Projektdatei) oder eine andere Dateiart ausgegeben. Der Quellcode wird anschließend in eine exe-Datei kompiliert. Nach Starten der exe-Datei greift diese auf die durch die Schema-Datei beschriebenen XML-Daten zu.

SPL-Dateien haben Zugriff auf die verschiedensten Informationen, die aus den Quellschemas geholt werden. Beachten Sie bitte, dass eine SPL-Datei nicht an ein bestimmtes Schema gebunden ist, sondern Zugriff auf alle Schemas erlaubt! Stellen Sie sicher, dass Sie Ihre SPL-Dateien generisch anlegen, Strukturen usw. vermeiden, die nur auf bestimmte Schemas zutreffen!

Erstellen einer neuen Datei in SPL:

```
[create "test.cpp"]
#include "stdafx.h"
[close]
```

Dies ist ein sehr einfacher SPL-Code. Damit wird eine Datei names **test.cpp** erstellt. Das Include Statement wird innerhalb davon positioniert. Der Befehl "close" schließt und beendet die Vorlage.

12.4.1 Code-Blöcke

Code Generator-Anweisungen sind in eckige Klammern eingeschlossen '**[**' und '**]**'.

In ein Klammerpaar können mehrere Anweisungen gesetzt werden. Zusätzliche Anweisungen müssen durch eine neue Zeile oder einen Doppelpunkt '**:**' getrennt werden.

Gültige Beispiele sind:

```
[$x = 0
$x = $x + 1]
```

or

```
[$x = 0: $x = $x +1]
```

12.4.2 Kommentare

Kommentare beginnen immer mit den Zeichen **'** und enden an der nächsten Zeile oder an einem Zeichen, mit dem ein Block geschlossen wird **]**.

© 2005 Altova GmbH

12.4.3 Variablen

Das **$-**Zeichen wird zur **Deklarierung** oder **Verwendung** einer Variable verwendet. Vor der Variable muss immer ein **$** stehen.

Variablentypen:

- integer
- string
- class iterator (siehe foreach-Anweisung)

Variablentypen werden nach der ersten Verwendung deklariert.

```
[$x = 0]
x, ist nun ein Integer

[$x = "teststring"]
x, wird nun als String behandelt
```

Arbeiten mit Strings:
Im oben gezeigten Beispiel wird gezeigt, wie ein fixed String einer Variable zugewiesen wird. Sie werden Strings außerdem verketten und sie anderen Variablen zuweisen müssen.

Zuweisen eines Strings zu einer Variable:

```
[$x = $module]
```

Zum Definieren einer String-Verkettung wird das **&-**Zeichen verwendet.

```
[$x = "testconcat" & "ination" & $module]
```

12.4.4 Globale Objekte

Nachdem eine Schema-Datei von Code Generator analysiert wurde, sind die in der Tabelle aufgelisteten Objekte im Template-Prozessor vorhanden.

Name	Typ	Beschreibung
$namespaces	Namespace-Sammlung	Sammlung von Namespace-Objekten
$classes	Class-Sammlung	Alle complexTypes, Elemente,… in einer Ansicht im flachen Format. Die Reihenfolge entspricht der beim Parsen durch die Typen ermittelten Reihenfolge.
$module	String	Name des Quellschemas ohne Erweiterung
$outputpath	String	Der vom Benutzer definierte Ausgabepfad oder der Standard-Ausgabepfad

12.4.5 Verwendung von Dateien

[create filename]
Erstellt eine neue Datei. Die Datei muss mit der **[close]**-Anweisung geschlossen werden.

[append filename]
Hängt Datenelemente an eine bestimmte Datei an. Die Datei muss mit der **[close]-**Anweisung geschlossen werden.

12.4.6 Bedingungen

Hinzufügen von Bedingungen zu Ihrem Code (if):
SPL erlaubt die Verwendung von "if"-Anweisungen - die Syntax lautet wie folgt:

```
[if  condition]
    [instruction block]
[else]
    [instruction block]
[endif]
```

oder, **ohne else**:

```
[if  condition]
    [instruction block]
[endif]
```

Beachten Sie bitte, dass für die Bedingung keine runden Klammern verwendet werden!

Bedingungen:
Wie bei jeder anderen Programmiersprache, werden Bedingungen mit logischen und Vergleichsoperatoren erstellt:

Logische Operatoren:

Operator	**Operator in SPL**
Not	not
And	and
Or	or

Vergleichsoperatoren:

Not equal	<>
Equal	=
Greater than	>
Greater than or equal	>=
Less than	<
Less than or equal	<=

Eine Beispiel-**if-**Anweisung mit einer Bedingung sieht etwa folgendermaßen aus:

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
    was immer Sie wollen ['fügt was immer Sie wollen in die erzeugte Datei ein]
[endif]
```


© 2005 Altova GmbH

12.4.7 foreach

Vektoren und Iteratoren

Ein Vektor ist eine **Sammlung** von Objekten - wie ein gewöhnliches Array. Iteratoren lösen das Problem des Speicherns und Inkrementierens von Array Indizes beim Aufrufen von Objekten.

foreach-Beispiel:

```
[foreach $class in $classes]
    [if not $class.IsInternal]
       class [=$class.Name];
    [endif]
[next]
```

Die erste Zeile:
$classes ist das **globale Objekt** von allen Typen. $classes ist auch ein Vektor - eine Sammlung von einzelnen Klassenobjekten.

Foreach iteriert durch alle Datenelemente in $classes und führt den Code innerhalb der Anweisung aus.

In jeder Iteration wird **$class** dem nächsten Klassenobjekt zugewiesen. Sie arbeiten einfach mit dem Klassenobjekt, anstelle classes[i]->class->Name(), zu verwenden, wie das in C++ der Fall wäre.

foreach wird bis zur **[next]-**Anweisung ausgeführt.

12.4.8 Hinzufügen von Text zu Dateien

Durch das Iterieren durch Sammlungen und die Inkrementierung von Werten werden keine kompilierbaren Dateien generiert.

Text, der nicht innerhalb von **[** und **]** steht, wird direkt in die Datei geschrieben. Der Vorgang ist ähnlich wie bei der Verwendung von Javascript in ASP.

Wenn Sie den Wert einer Variable in das erzeugte Dokument platzieren wollen, verwenden Sie die folgende Methode:

```
Das Ergebnis Ihrer Berechnung ist [=$x] - einen schönen Tag noch…
```

- Angenommen der Wert der Variable x ist 23, dann lautet die erzeugte Datei:

```
Das Ergebnis Ihrer Berechnung ist 23 - einen schönen Tag noch…
```

12.4.9 Subroutinen

Code Generator unterstützt nun Subroutinen in der Form von Prozeduren.

Funktionen:

- Übergabe von Werten nach Wert und nach Referenz
- Lokale/globale Parameter (lokal innerhalb von Subroutinen)
- Rekursiver Aufruf (Subroutinen können sich selbst aufrufen)

Deklaration einer Subroutine

Subroutinen
Subroutinen sind derzeit als **Prozeduren** implementiert. Funktionen werden nicht unterstützt.

Syntax:

```
Sub SimpleSub()
... lines of code
EndSub
```

- **Sub** ist das Schlüsselwort, das die Prozedur notiert.
- **SimpleSub** ist der Name, der der Subroutine zugewiesen wurde.
- Runde **Klammern** können eine Parameterliste enthalten.
- Der Code-Block einer Subroutine beginnt direkt nach der geschlossenen Klammer nach dem Parameter.
- **EndSub** notiert das Ende des Code-Blocks.

Bitte beachten Sie:
Eine rekursive oder kaskadierende Subroutine-**Deklaration** ist nicht zulässig, d.h. eine Subroutine darf keine weiteren Subroutinen enthalten.

Parameter:
Parameter können auch durch Prozeduren unter Verwendung der folgenden Syntax übergeben werden:

- Alle Parameter müssen Variablen sein
- Variablen muss das **$-**Zeichen vorangestellt werden
- Lokale Variablen werden in einer Subroutine definiert
- Globale Variablen werden explizit außerhalb von Subroutinen deklariert
- Mehrere Parameter werden innerhalb runder Klammern durch Kommas "**,**" voneinander getrennt
- Parameter können Werte übergeben

Parameter - Übergabe von Werten
Parameter können auf zwei Arten übergeben werden: nach Wert oder nach Referenz, wobei die Schlüsselwörter ByVal bzw. ByRef verwendet werden.

Syntax:

```
' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
] ...
```

- ByVal definiert, dass der Parameter nach Wert übergeben wird.
- ByRef definiert, dass der Parameter nach Referenz übergeben wird.

Subroutinenaufruf

Verwenden Sie zum Aufrufen einer Subroutine **call**, gefolgt vom Namen der Prozedur und ggf. den Parametern.

```
Call SimpleSub()
```

oder,

```
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
```


© 2005 Altova GmbH

Zulässige Parameterzeichen:
Variablen, Strings, Zahlen, true und false sind die einzigen Zeichen, die derzeit innerhalb von Parametern verwendet werden können. Verwenden Sie keine Ausdrücke!

Beispiel für eine Subroutine

Das markierte Beispiel enthält die Deklaration und den Aufruf einer Subroutine.

A sample SPL file:

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
call SimpleSub()

$ParamByValue    = "Original Value"
]ParamByValue    = [=$ParamByValue]
[$ParamByRef     = "Original Value"
]ParamByRef      = [=$ParamByRef]

' define sub CompleteSub()
[sub CompleteSub( $param, byval $paramByValue, byref $paramByRef )
]CompleteSub called.
    param = [=$param]
    paramByValue = [=$paramByValue]
    paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]   Set values inside sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
close
]
```

Derselbe Beispielcode:

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
Sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
Call SimpleSub()
```

```
$ParamByValue       = "Original Value"
]ParamByValue       = [=$ParamByValue]
[$ParamByRef  = "Original Value"
]ParamByRef   = [=$ParamByRef]

' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
]CompleteSub called.
       param = [=$param]
       paramByValue = [=$paramByValue]
       paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]      Set values inside Sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
Close
]
```

12.4.10 Integrierte Typen

In diesem Abschnitt werden die in globalen Variablen verwendeten integrierten Typen beschrieben, die das geparste Schema beschreiben.

Namespace

Namespace-Abstraktion:

Name	Typ	Beschreibung
URI	string	Die URI dieses Namespace
Prefix	string	Das Präfix dieses Namespace
ContainsPublicClasses	boolean	True, wenn die Klassensammlung mindestens ein nicht-internes Klassenobjekt enthält.
Classes	Class collection	Sammlung von Klassen in diesem Namespace - iterieren Sie mit **foreach** durch diese

Klasse

Klassen-Abstraktion:

© 2005 Altova GmbH

Name	Typ	Beschreibung
HasNamespace	boolean	True, wenn es ein verknüpftes Namespace-Objekt gibt.
Namespace	Namespace	Das Namespace-Objekt, dem dieses Klassenobjekt angehört.
NamespacePrefix	string	Das Präfix des Namespace, in dem sich die Klasse befindet.
NamespaceURI	string	URI des Namespace, in dem sich die Klasse befindet.
Name	string	Name des Typs der erzeugten Datei
SchemaName	string	Name des Typs in der Original-Schema-Datei
HasBaseObject	boolean	True, wenn dieser Typ von einem anderen Typ abgeleitet wurde, der auch durch ein Klassenobjekt dargestellt wird.
BaseObject	Class	Das Basisklassenobjekt, wenn HasBaseObject "true" ist.
BaseNamespaceURI	string	Namespace URI der Basisklasse
Base	string	Name der Basisklasse
SchemaBase	string	Name des Basistyps in der Original-Schema-Datei
BuiltInBase	string	Name des Root Simple Type
BuiltInSchemaBase	string	Name des Root Simple Type in der Original-Schema-Datei
IsRoot	boolean	True, wenn ein Root-Element dieses Typs vorhanden ist.
IsSimpleType	boolean	True, wenn dies ein simple Type ist.
IsComplexFromSimpleType	boolean	True, wenn dies ein complex Type ist, der von einem simple Type abgeleitet wurde.
IsComplexType	boolean	True, wenn ein complex Type vorhanden ist.
IsSequence	boolean	True, wenn der Group-Type der obersten Ebene "sequence" ist.
IsChoice	boolean	True, wenn der Group-Type der obersten Ebene "choice" ist.
IsAll	boolean	True, wenn der Group-Type der obersten Ebene "all" ist.
Description	string	Beschreibung dieses Typs. Kann Zeilenvorschubzeichen enthalten.
IsGlobal	boolean	True, wenn dieser Typ überall in den generierten Dateien verwendet werden kann.
IsAnonymous	boolean	True, wenn dieser Typ eine Art von Help-Objekt ist, das in den generierten Dateien nicht angezeigt wird.
IsInternal	boolean	True, wenn dies ein Help-Objekt ist, das Root-Elemente behandelt.
Members	Member collection	Eine Klasse, die einen complex Type darstellt, enthält ein oder mehrere Members.
Facets	Facet collection	Eine Klasse, die einen simpleType oder einen von einem simpleType abgeleiteten complexType darstellt, kann Facets enthalten.

Member

Abstraktion einer Member-Variable innerhalb einer Klasse.

Name	Typ	Beschreibung
NamespaceURI	string	Die Namespace URI dieses Elements/Attributs innerhalb von XML-Instanzdokumenten/Streams.
Name	string	Name in der erzeugten Datei
SchemaName	string	Name, wie er in der Original-Schema-Datei aufscheint.
XmlName	string	Der Name, wie er in den XML-Instanzdokumenten/Streams aufscheinen soll.
Type	string	Der Name der Klasse, die den Schematyp darstellt.
SchemaType	string	Der Schematyp.
TypeObject	Class	Siehe Erklärung unten*****
HasTypeObject	boolean	True, wenn typeobject einen gültigen Wert hat.
Description	string	Beschreibung des Elements/Attributs. Kann Zeilenvorschubzeichen enthalten.
IsBuiltInType	boolean	True, wenn der Typ ein integrierter Schematyp wie z.b. string, unsignedInt, dateTime... ist.
IsSimpleType	boolean	True, wenn der Typ dieses Member ein simpleType ist.
IsElement	boolean	True, wenn dies ein Element ist.
IsAttribute	boolean	True, wenn dies ein Attribut ist.
NodeType	string	"Element" oder "Attribute"
MinOcc	integer	minOccurs, wie in Schema
MaxOcc	integer	maxOccurs, wie in Schema
IsQualified	boolean	True, wenn die Form des Elements/Attributs auf "qualified" gesetzt ist.

***TypeObject:**
Dies muss aus folgendem Grund verwendet werden: Sie könnten einen complexType verwenden, wobei der complexType selbst einen simpleType enthält.

- Holen Sie die $class-Variable und iterieren Sie mittels **foreach** durch die Members - bis Sie den simpleType erreichen.
- Sie benötigen die Eigenschaften des simpleType: type name oder andere beliebige Eigenschaften.
- Überprüfen Sie **HasTypeObject**. Wenn der Wert true ist, wurde das Member mit gültigem Inhalt befüllt und kann geparst werden.

© 2005 Altova GmbH

Facet

Diese Klasse besteht aus dem Constraint selbst und mehreren Booleschen Variablen.

Die Booleschen Variablen geben an, um welche Art von Constraint es sich in dem Fall handelt. Die Namen der Booleschen Variablen sind mit den in der Schema-Spezifikation angegebenen identisch.

Name	Typ	Beschreibung
Constraint	string	Enthält den Wert des Facet.
IsLength	boolean	
IsMinLength	boolean	
IsMaxLength	boolean	
IsMinInclusive	boolean	
IsMinExclusive	boolean	
IsMaxInclusive	boolean	
IsMaxExclusive	boolean	
IsTotalDigits	boolean	
IsFractionDigits	boolean	
IsWhiteSpace	boolean	
IsPattern	boolean	
IsEnumeration	boolean	
Enumeration	Enumeration collection	Enthält eine Sammlung von Enumerationsobjekten, wenn dieses Facet dem Typ Enumeration angehört.

Enumeration

Abstraktion eines Enumerationseintrags innerhalb eines Facet.

Name	Typ	Beschreibung
Index	integer	Enthält den Index dieses Enumerationswerts, beginnend mit 1.
Value	string	Enthält einen Enumerationswert.

12.5 Fehlercodes

Betriebssystem-Fehlercodes
201 Datei nicht gefunden: '%s'
202 Datei '%s' kann nicht erstellt werden
203 Datei '%s' konnte nicht geöffnet werden

Schema-Fehlercodes
301 Grammatikdatei konnte nicht initialisiert werden
302 Validator: %s
303 Validator kann Schema '%s' nicht laden
304 Validator kann nicht generiert werden

Syntax-Fehlercodes
401 Keyword erwartet
402 '%s' erwartet
403 Es wurde keine Output-Datei angegeben

Runtime-Fehlercodes
501 Unbekannte Variable '%s'
502 Umdefinition der Variablen '%s'
503 Variable '%s' ist kein Container
504 Unbekannte Eigenschaft '%s'
505 Kann %s nicht in %s konvertieren

© 2005 Altova GmbH

13 Benutzerreferenz

Im Referenzteil der Dokumentation finden Sie eine Beschreibung aller XMLSpy 2005 Menübefehle und deren Verwendung. Wir haben versucht, das Handbuch so umfassend wie möglich zu gestalten. Sollten Sie trotzdem Fragen zu Themen haben, die in der vorliegenden Dokumentation nicht behandelt werden, werfen Sie bitte einen Blick in die FAQs und die Diskussionsforen auf der Altova Website. Sollten Ihre Fragen damit noch immer nicht beantwortet sein, kontaktieren Sie uns bitte über unser Support Center auf der Altova Website.

Bitte beachten Sie, dass im Menü Datei und Bearbeiten alle Windows-Standardbefehle und alle zusätzlichen Befehle im Zusammenhang mit XML- und Internetfunktionalitäten unterstützt werden.

13.1 Menü "Datei"

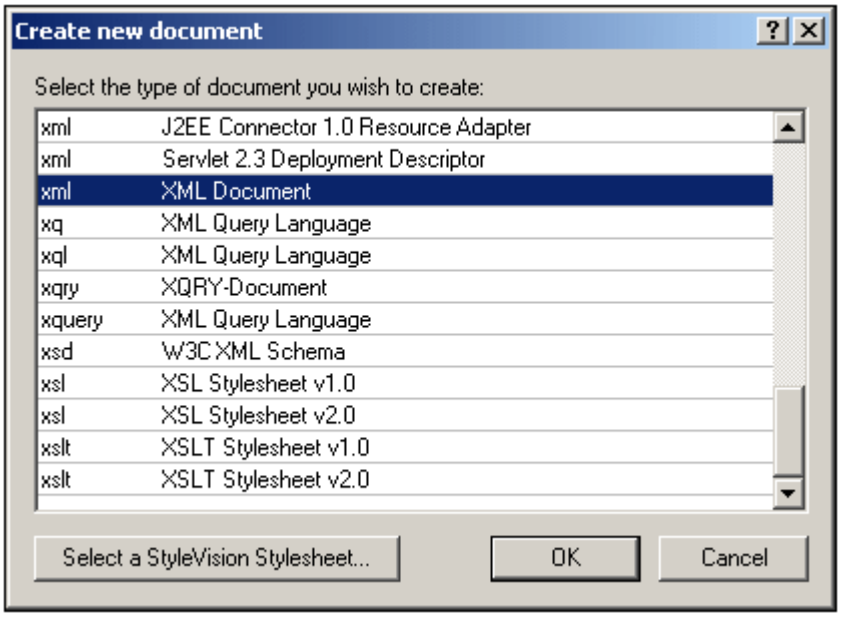
Das Menü "Datei" enthält alle Befehle, die zum Arbeiten mit Dateien benötigt werden. Die Befehle sind in der aus Windows Software-Produkten bekannten Reihenfolge angeordnet.

Zusätzlich zu den Standardbefehlen Neu, Öffnen, Speichern, Drucken, Druckereinrichtung, und Beenden bietet XMLSpy 2005 eine Reihe von XML- und applikationsspezifischen Befehlen.

13.1.1 Neu...

Strg+N

Der Befehl **Neu...** wird verwendet, um ein neues Dokument zu erstellen. Wenn Sie auf Neu… klicken, wird das Dialogfeld "Neues Dokument anlegen" geöfffnet, in dem Sie auswählen können, welche Art von Dokument erstellt werden soll. Falls das Dokument, das Sie erstellen wollen, nicht aufgelistet ist, wählen Sie "XML", und ändern Sie die Erweiterung des Dokuments beim Speichern. Beachten Sie, dass Sie neue Dateiarten in die Liste einfügen können, indem Sie diese unter Extras | Optionen auf dem Register "Dateiarten" eintragen.

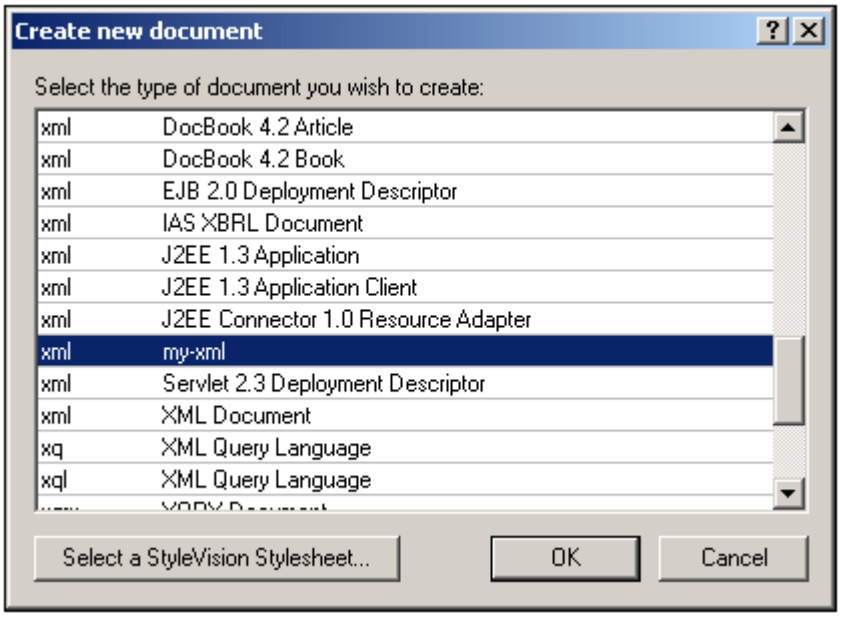


Erstellen von Vorlagen für neue Dokumente
Sie können mehrere Vorlagen für verschiedene Dateiarten erstellen, die Sie anschließend direkt über das Dialogfeld "Neues Dokument anlegen" öffnen und bearbeiten können. Um Ihre eigene Vorlage zu erstellen, sodass Sie in diesem Dialogfeld in der Liste der Dokumente angezeigt wird, müssen Sie zuerst ein Vorlagendokument erstellen und dieses dann in dem Ordner, der alle Vorlagen enthält, speichern. Gehen Sie vor wie folgt:

1. Öffnen Sie den Ordner **`XMLSpy 2005\Template`** über den Windows Explorer oder Ihr bevorzugtes Navigationstool und wählen Sie aus den mit **`new.xxx`** benannten Dateien (wobei `.xxx` eine Dateierweiterung wie z.B. `.xml` und `.xslt` ist) eine rudimentäre Vorlagendatei aus.
2. Öffnen Sie die Datei in XMLSpy 2005 und ändern Sie die Datei Ihren Wünschen entsprechend. Diese Datei wird dann als Vorlage verwendet.
3. Wählen Sie anschließend **Datei | Speichern unter...** um die Datei wieder im Ordner

 © 2005 Altova GmbH

\Template unter einem passenden Namen z.B. my-xml.xml zu speichern. Sie haben nun eine Vorlage mit dem Namen `my-xml.xml`, die im Dialogfeld "Neues Dokument anlegen" in der Dateiliste angezeigt wird.

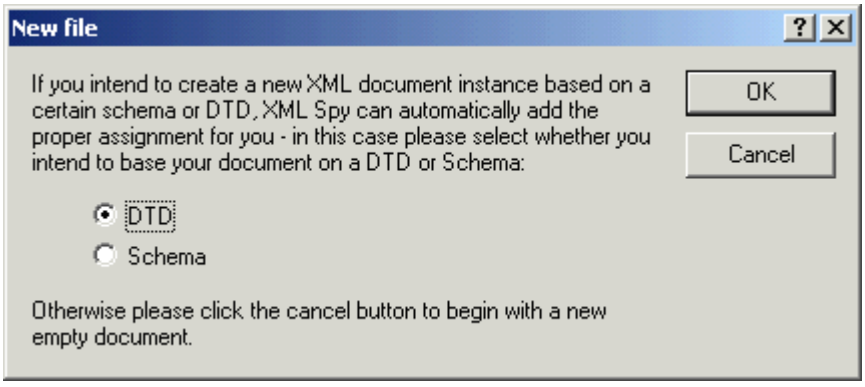


4. Zum Öffnen der Vorlage wählen Sie **Datei | Neu** und anschließend die Vorlage (in diesem Fall `my-xml`).

Anmerkung: Um eine Vorlage zu löschen, öffnen Sie das Vorlagen-Verzeichnis und löschen Sie dort die gewünschte Vorlage.

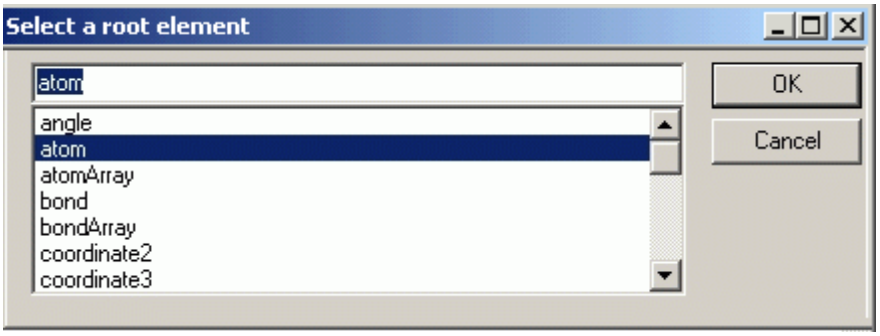
DTD / XML-Schema einem neuen XML-Dokument zuweisen
Wenn Sie ein neues Dokument einer bestimmten Art erstellen, das auf einem Standardschema (DTD oder XML-Schema) basiert, wird das Dokument automatisch mit der korrekten DTD oder XML-Schema-Verknüpfung geöffnet. So wird z.B. eine XHTML-Datei mit der damit verknüpften DTD `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd` geöffnet. Eine XML-Schema-Datei (`.xsd`) wird mit dem Schemadokument `http://www.w3.org/2001/XMLSchema` verknüpft.

Wenn Sie eine neue Datei erstellen, für die das Schema nicht bekannt ist (z.B. eine XML-Datei), werden Sie aufgefordert, ein Schema (DTD oder XML-Schema) mit dem zu erstellenden Dokument zu verknüpfen.



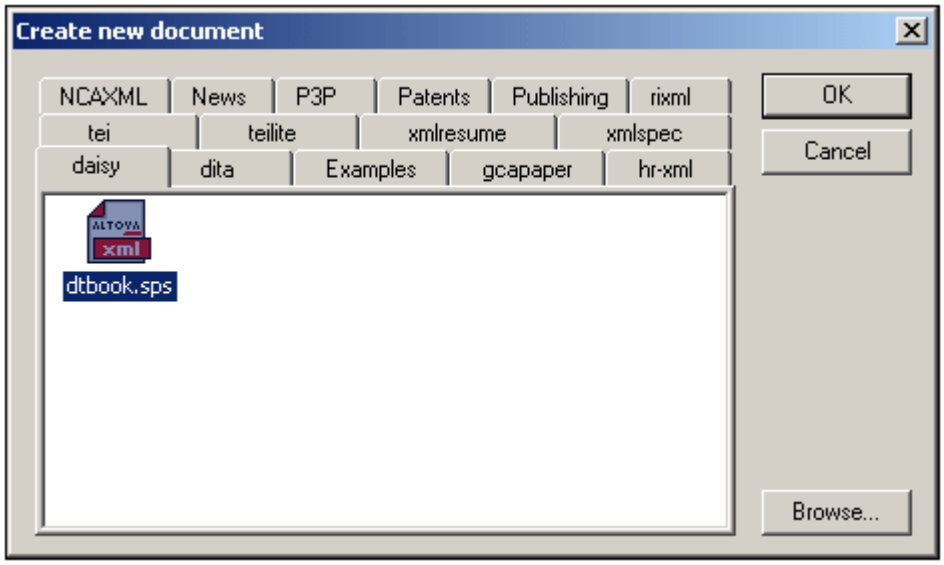
Wenn Sie eine DTD oder ein XML-Schema mit Ihrem Dokument verknüpfen, klicken Sie im Dialogfeld "Neue Datei" auf OK, um den Pfad zum Schema anzugeben. Wenn Sie in diesem Dialogfeld auf "Abbrechen" klicken, wird eine neue Datei erstellt, die mit keinem Schema verknüpft ist.

Angabe des Dokumentelements eines neuen XML-Dokuments
Wenn Sie ein XML-Schema auswählen, kann dieses mehr als ein globales Element enthalten. Jedes davon kann potenziell ein Dokument- (oder Root) Element sein. Im Dialogfeld "Wählen Sie das Root-Element" können Sie das Root-Element für das XML-Dokument auswählen. Dieses Dialogfeld wird angezeigt, wenn Sie im Dialogfeld "Neue Datei" Schema auswählen und wenn das XML-Schema mehrere globale Elemente enthält.



Das neue XML-Dokument wird mit diesem Element als Dokument-Element erstellt.

Zuweisen eines StyleVision Power Stylesheet beim Erstellen eines neuen Dokuments
Bei Erstellung eines neuen XML-Dokuments können Sie es mit einem StyleVision Power Stylesheet (`.sps`-Datei) verknüpfen, um es in der Authentic-Ansicht anzeigen zu können. Wenn Sie im Dialogfeld "Neues Dokument anlegen" (*siehe Abbildung oben*) auf die Schaltfläche "StyleVision Stylesheet auswählen" klicken, erscheint das Dialogfeld "Neues Dokument anlegen" (*siehe Abbildung unten*).



Auf den verschiedenen Ordner-Registern dieses Dialogfelds stehen Ihnen die gewünschten StyleVision Power Stylesheets zur Verfügung. Alternativ dazu können Sie das gewünschte StyleVision Power Stylesheet auch über die Schaltfläche **Durchsuchen…** suchen. Die Register in diesem Dialogfeld entsprechen den Ordnern im Ordner `sps/Template` im Programmordner

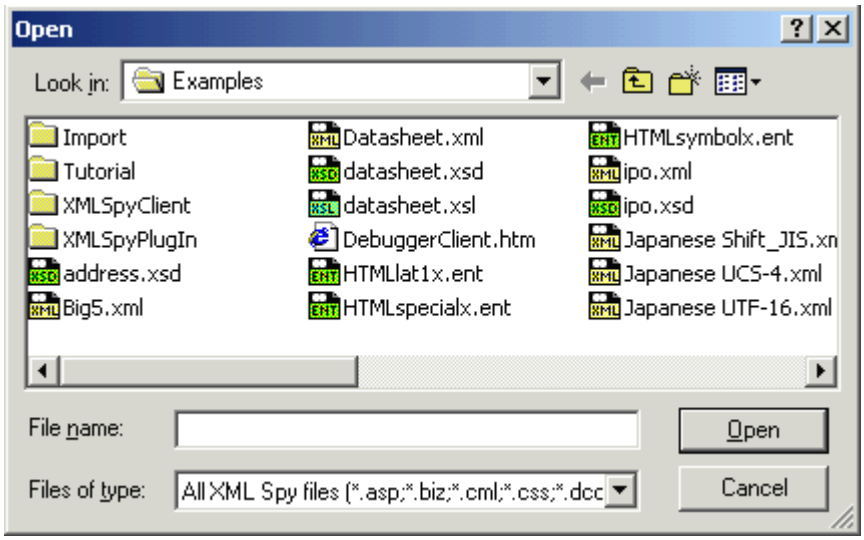
 © 2005 Altova GmbH

der Applikation.

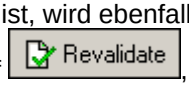
13.1.2 Öffnen...

Strg+O

Mit "Öffnen..." rufen Sie das bekannte Windows-Dialogfeld "Öffnen" (*siehe Abbildung unten*) auf, über das Sie jedes XML-Dokument oder jede Textdatei öffnen können. Sie haben auch die Möglichkeit, mehrere Dateien auszuwählen. Im Listenfeld "Dateityp" können Sie auswählen, welche Dateien im Dialogfeld angezeigt werden sollen (die Liste der verfügbaren Dateiarten kann auf dem Register "Dateiarten" unter Extras | Optionen eingestellt werden).



Beim Öffnen einer XML-Datei wird diese auf Wohlgeformtheit überprüft. Ist sie nicht wohlgeformt, wird eine Fehlermeldung angezeigt. Beheben Sie den Fehler und klicken Sie auf

, um die Datei erneut zu überprüfen. Wenn Sie die Option Validierung beim Öffnen der Datei aktiviert haben und die Datei ungültig ist, wird ebenfalls eine Fehlermeldung angezeigt. Beheben Sie den Fehler und klicken Sie auf 

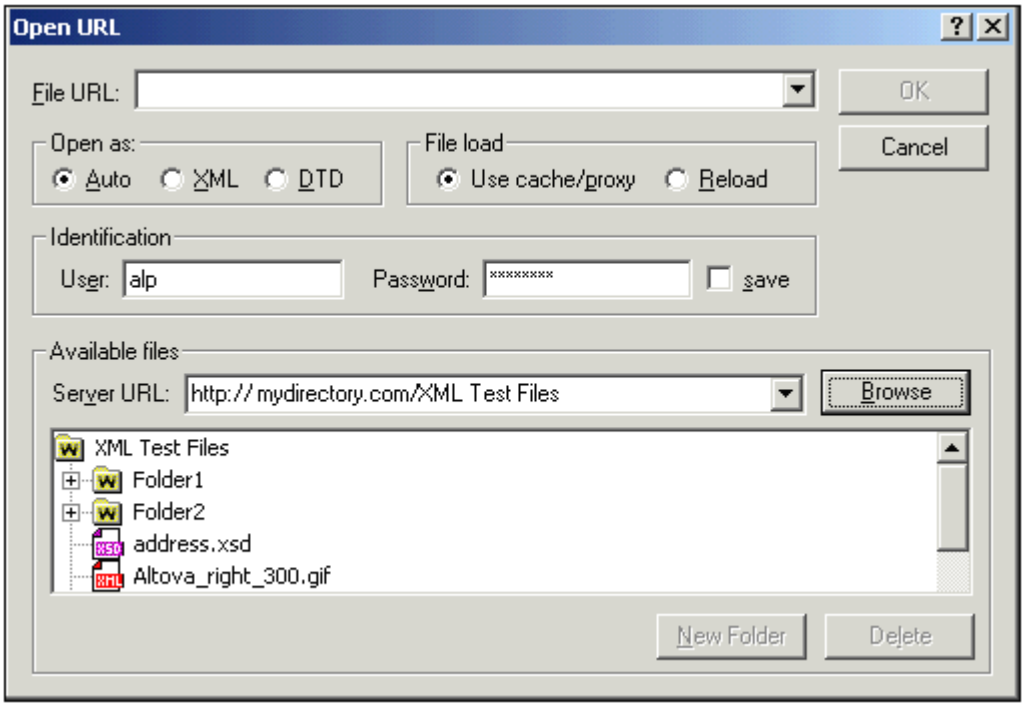
, um das Dokument erneut zu validieren.

13.1.3 URL öffnen...

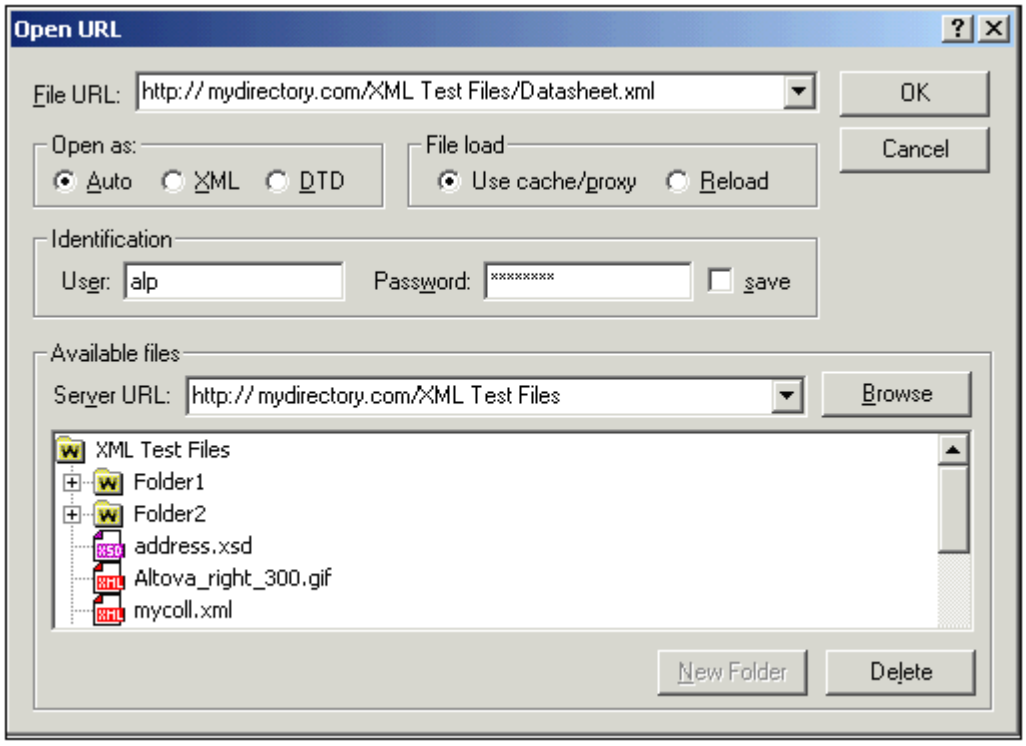
Mit **URL öffnen...** können Sie Dateien, die nicht lokal abgelegt sind, aus einer URL (Uniform Resource Locator) mittels Http und WebDAV öffnen.

So öffnen Sie eine URL Adresse:

1. Klicken Sie auf den Befehl "URL öffnen". Daraufhin wird das Dialogfeld "URL öffnen" angezeigt.



2. Geben Sie die URL im Feld "Server URL" ein.
3. Falls der Server passwortgeschützt ist, geben Sie Ihren Benutzernamen und Ihr Passwort in den entsprechenden Feldern ein.
4. Klicken Sie auf die Schaltfläche **Durchsuchen** um durch die Verzeichnisstruktur des Servers zu navigieren.
5. Klicken Sie auf die in XMLSpy 2005 zu ladende Datei.



Im Feld "Datei-URL" wird die URL der Datei angezeigt. Jetzt ist auch die Schaltfläche

 © 2005 Altova GmbH

OK aktiv.

5. Klicken Sie auf **OK** um die Datei zu laden.
   Die Datei wird nun im Hauptfenster angezeigt.

Hinweis: Die Funktion **Durchsuchen** ist nur auf Servern, die die Protokolle FTP, HTTP und HTTPS (bei Unterstützung von WebDAV) und auf Servern, die WebDAV unterstützen, verfügbar.

Datei laden
Um Ihnen mehr Kontrolle über den Ladeprozess zu geben, können Sie die Datei über den lokalen **Cache** oder den **Proxyserver** laden (was den Prozess beschleunigt, wenn die Datei zuvor schon einmal geladen worden ist). Alternativ dazu können Sie die Datei, wenn Sie z.B. mit einem Datenbanksystem oder einem Electronic Publishing-System arbeiten, über die Option "Neu laden" auch erneut laden.

Speicherschnittstellen
XMLSpy 2005 unterstützt drei Arten von Speicherschnittstellen:

- Generische Versionskontrollsysteme wie Microsoft Visual Source-Safe (oder kompatible Produkte)
- Generische Webserver wie WebDAV und FTP
- Spezielle XML- oder Schema-Speicherschnittstellen, wie Tamino, TEXTML Server, Virtuoso oder XML Canon.

XMLSpy 2005 unterstützt derzeit Source-Safe, FTP und WebDAV Server sowie Virtuoso über WebDAV. Zukünftige Versionen werden weitere Schnittstellen unterstützen.

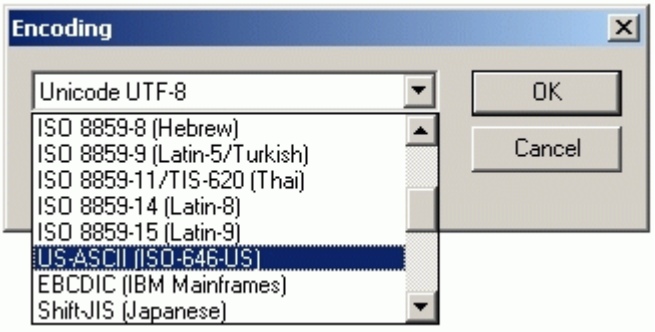
13.1.4 Neu laden

Mit **Neu laden** können Sie geöffnete Dokumente erneut laden.

Dieser Befehl ist nützlich, wenn ein geöffnetes Dokument außerhalb von XMLSpy 2005 geändert wurde. In diesem Fall werden Sie von XMLSpy 2005 gefragt, ob die Datei erneut geladen werden soll. Wenn Sie dies tun, gehen alle Änderungen, die Sie an der Datei seit der letzten Speicherung vorgenommen haben, verloren. Sie können diese Option im Dialogfeld "Optionen" (Extras | Optionen) ändern.

13.1.5 Kodierung...

Mit **Kodierung...** können Sie die Kodierung der in Verwendung befindlichen Datei ansehen und beim nächsten Speichern ändern.



Wenn Sie eine andere Kodierung als die derzeit verwendete auswählen, wird die Kodierungsspezifikation der XML-Deklaration entsprechend geändert. Bei 16-bit und 32-bit pro Zeichen (UTF-16, UCS-2 und UCS-4) können Sie weiters die zu verwendende Bytefolge in der Datei angeben. Sie können die neue Kodierung auch in der Kodierungsspezifikation der XML-Deklaration angeben. Beim Speichern eines Dokumentes prüft XMLSpy 2005 automatisch die Kodierungsspezifikation und öffnet ein Dialogfeld, falls es den vom Benutzer angegebenen Kodierungsnamen nicht erkennt.

Hinweis: Sollte Ihr Dokument Zeichen enthalten, die in der gewählten Kodierung nicht angezeigt werden können, erhalten Sie eine Warnmeldung beim Speichern der Datei.

13.1.6 Schließen

Mit **Schließen** beenden Sie das aktive Dokument. Sollten Sie die Datei verändert haben (in der Titelleiste wird der Dateiname mit einem Sternchen "*" angezeigt), werden Sie gefragt, ob Sie die Datei vor dem Schließen speichern möchten.

13.1.7 Alle schließen

Mit **Alle Schließen** beenden Sie alle Dokumente. Sollten Sie Dateien verändert haben (in der Titelleiste wird der Dateiname mit einem Sternchen "*" angezeigt), werden Sie gefragt, ob Sie die Dateien vor dem Schließen speichern möchten.

13.1.8 Speichern

Strg+S

Mit **Speichern** wird der Inhalt des aktiven Dokuments in der geöffneten Datei gespeichert. Beim Speichern wird das Dokument automatisch auf Wohlgeformtheit geprüft. Außerdem wird das Dokument auch automatisch validiert, wenn diese Option im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" aktiviert wurde. Die XML Deklaration wird auch auf Kodierungsspezifikationen geprüft. Diese Kodierung wird beim Speichern des Dokuments verwendet.

13.1.9 Speichern unter...

Mit dem Befehl **Speichern unter** gelangen Sie zu dem aus Windows bekannten Dialogfeld "Speichern unter...". Hier geben Sie den Namen und den Ordner an, in dem die Datei gespeichert werden soll. Bei Verwendung des Befehls **Speichern unter** werden dieselben Validierungen und Prüfungen durchgeführt, wie bei dem Befehl **Speichern.**

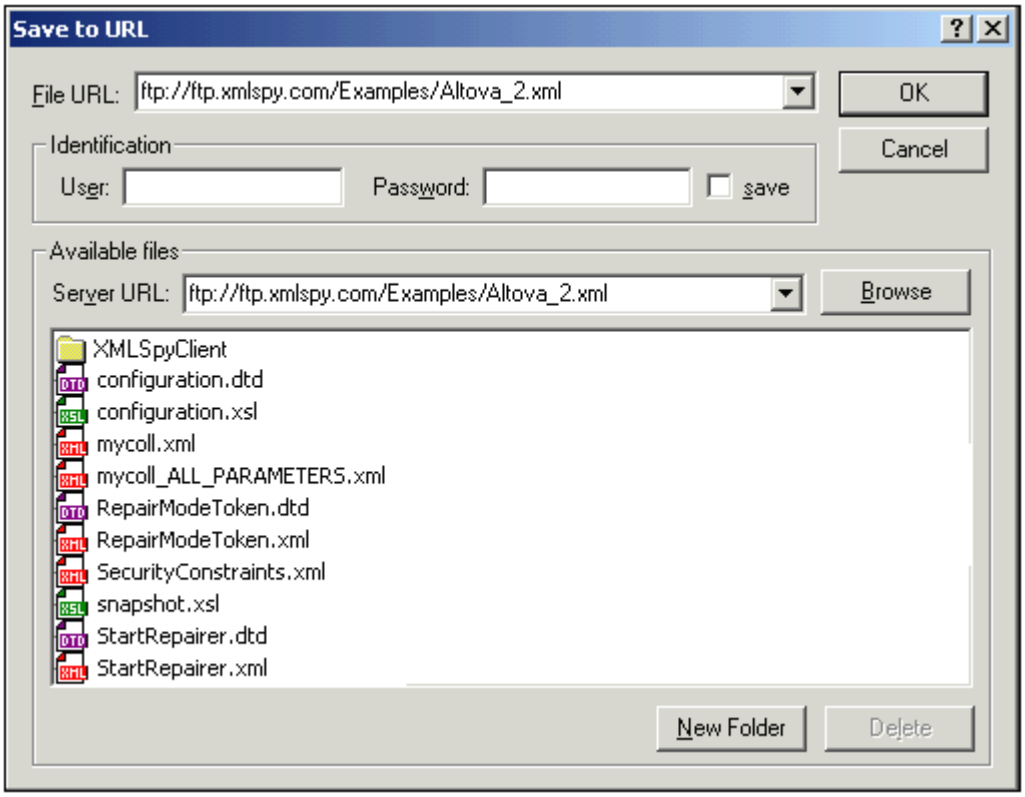
13.1.10 Speichern unter URL...

Mit "Speichern unter URL..." können Sie Dateien, die nicht lokal abgelegt sind, unter einer URL-(Uniform Resource Locator) Adresse speichern.

So speichern Sie ein XML-Dokument unter einer URL-Adresse:

1. Die zu speichernde Datei muss im Hauptfenster aktiv sein.
2. Wählen Sie die Menüoption **Datei | Speichern unter URL.**

© 2005 Altova GmbH



3. Klicken Sie auf die Schaltfläche **Durchsuchen** um durch die Verzeichnisstruktur auf dem Server zu navigieren.
4. Geben Sie den Namen im Feld Datei-URL ein, oder markieren Sie einen Eintrag im Listenfeld "Vorhandene Dateien", wenn Sie die Datei überschreiben möchten.

Hinweis:

- Die Funktion **Durchsuchen** ist nur für Server verfügbar, die folgende Protokolle unterstützen: FTP, HTTP und HTTPS (falls der Server WebDAV unterstützt) und auf Servern, die WebDAV unterstützen.
- Falls der Server, mit dem Sie sich verbinden möchten, passwortgeschützt ist, geben Sie bitte die benötigten Daten ein, und klicken Sie zum Verbinden auf die Schaltfläche **OK**. Sie können Ihre Daten auch in den Feldern "Benutzer" und "Passwort" eingeben. Mit dem Kontrollkästchen "Speichern" werden die Passwortdaten für spätere Logins gespeichert.

Speicherschnittstellen

XMLSpy 2005 unterstützt drei Arten von Speicherschnittstellen:

- Generische Versionskontrollsysteme wie Microsoft Visual Source-Safe (oder kompatible Produkte)
- Generische Webserver wie WebDAV und FTP
- Spezielle XML- oder Schema-Speicherschnittstellen, wie Tamino, TEXTML Server, Virtuoso oder XML Canon.

XMLSpy 2005 unterstützt derzeit Source-Safe, FTP und WebDAV Server, sowie Virtuoso über WebDAV. Zukünftige Versionen werden weitere Schnittstellen unterstützen.

13.1.11 Alles speichern

Mit diesem Befehl werden alle Änderungen gespeichert, die Sie an offenen Dokumenten durchgeführt haben. Dieser Befehl ist hilfreich, wenn Sie gleichzeitig an mehreren Dokumenten arbeiten. Falls ein zu speicherndes Dokument bisher noch nie gespeichert wurde, wird das Dialogfeld Speichern unter... (z.B. ein über den Befehl Neu erstelltes Dokument) für dieses Dokument angezeigt.

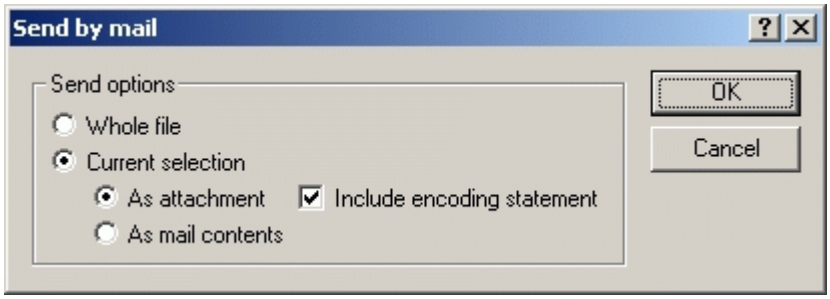
13.1.12 Als Mail senden...

Mit dem Befehl **Als Mail senden...** können Sie jede XML-Datei oder Teile einer XML-Datei als Email versenden. Sie haben folgende Möglichkeiten:

- Versenden eines XML-Dokuments als Anhang oder als Inhalt einer E-Mail.
- Versenden eines Teils eines XML-Dokuments als Anhang oder Inhalt einer E-Mail.
- Versenden einer Gruppe von Dateien (die im Projektfenster ausgewählt wurden) als Anhang einer E-Mail.
- Versenden einer URL (die im Projektfenster ausgewählt wurde) als Anhang oder als Link.

Hinweis: Zur Verwendung dieser Funktion benötigen Sie ein MAPI-kompatibles E-Mail-System.

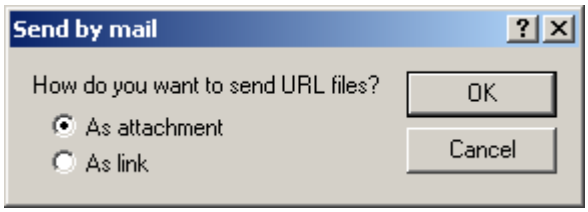
Versenden von Dokumenten und Teilen eines Dokuments
Um ein XML-Dokument zu versenden, machen Sie es im Haupfenster zum aktiven Dokument. Wenn Sie nur einen Teil aus einem XML-Dokument oder eine Gruppe von Dateien eines Projekts versenden möchten, markieren Sie den Abschnitt im Dokument oder wählen Sie die gewünschten Dateien im Projekt aus. Klicken Sie nun auf **Als Mail senden...**. Daraufhin wird das folgende Dialogfenster angezeigt.



Aktivieren Sie die gewünschten Optionen und klicken Sie auf **OK**. Das/die ausgewählte(n) Dokument(e)/der ausgewählte Inhalt/die ausgewählten URLs werden an die E-Mail angehängt bzw. der Inhalt wird in die E-Mail eingefügt.

URL per Mail versenden
Zum Versenden einer oder mehrerer URLs wählen Sie die URLs im Projektfenster aus und klicken Sie auf **Als Mail senden...**. Es wird das folgende Dialogfenster angezeigt.

© 2005 Altova GmbH



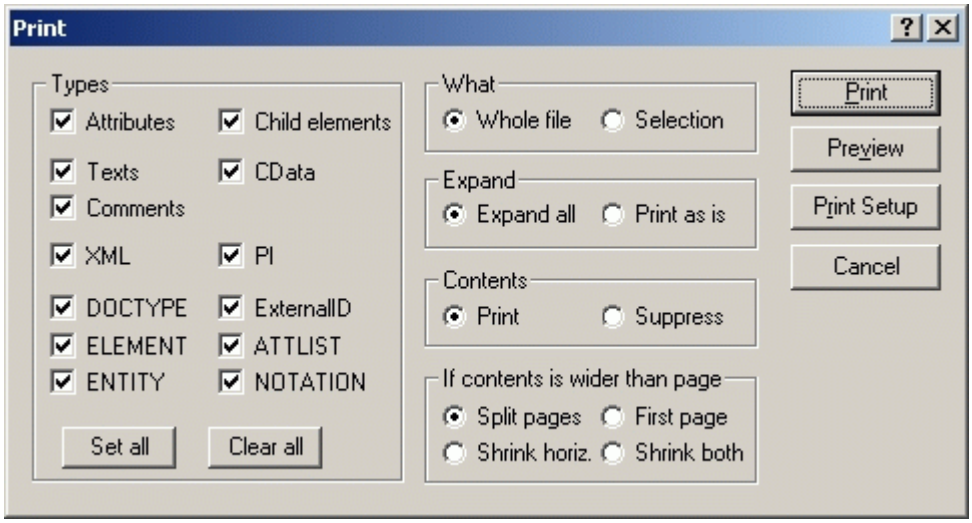
Wählen Sie aus, in welcher Form die URL versendet werden soll und bestätigen Sie mit **OK**.

13.1.13 Drucken...

Strg+P

Der Befehl **Drucken...** öffnet das Dialogfeld "Drucken". Hier können Sie festlegen, was die Druckausgabe enthalten soll.

Wenn Sie in der Enhanced Grid-Ansicht auf **Drucken...** klicken, wird ein Dialogfeld mit Druckoptionen geöffnet (*siehe Abbildung unten*), in dem Sie Druckoptionen für das XML-Dokument einstellen können. Durch Klicken auf die Schaltfläche "Drucken" wird das Dialogfeld für die Druckeroptionen geöffnet.



Im Folgenden sind die Optionen für das Drucken in der Enhanced Grid-Ansicht beschrieben:

- Im Bereich **Typ** haben Sie die Möglichkeit festzulegen, was in der Druckausgabe enthalten sein soll.
- Unter **Drucken** können Sie auswählen, ob die gesamte Datei oder ein ausgewählter Bereich gedruckt werden soll.
- Im Bereich **Erweitern** geben Sie an, ob das Dokument so gedruckt werden soll, wie es ist, oder ob Sie alle Child-Elemente ausgeklappt sehen möchten.
- Im Bereich **Inhalt** können Sie entscheiden, ob Sie den Inhalt aller Nodes oder nur die Node-Namen drucken möchten.
- Im Bereich **Wenn Inhalt breiter als Seite** können Sie entscheiden, was geschehen soll, wenn das Dokument größer als eine Papierseite ist. Die Option *aufteilen* druckt das gesamte Dokument in normaler Größe, und der Inhalt wird auf mehrere Seiten aufgeteilt (sowohl senkrecht als auch waagerecht). Sie können die Seiten dann zu einem Poster zusammenfügen. Die Option *Erste Seite* ermöglicht es Ihnen, nur die erste linke Seite des Druckbereichs zu drucken. Der restliche Bereich, der über den

Seitenrand ragt, wird nicht gedruckt. Diese Option eignet sich v.a. dann, wenn sich die meisten wichtigen Informationen in der Grid-Ansicht des Dokuments auf der linken Seite befinden. Die Option *Horizontal verkleinern* verkleinert die Druckausgabe bis sie horizontal auf der Papierseite Platz hat. Die Druckausgabe kann mehrere Seiten lang sein. Die Option *Beide verkleinern* reduziert die Größe der Druckausgabe bis sie ganz auf einer Papierseite Platz hat.

- Mit der Schaltfläche **Drucken** wird das Dokument nach Ihren Vorgaben ausgedruckt.
- Mit der Schaltfläche **Vorschau** erhalten Sie eine Vorschau auf die Druckausgabe, bevor diese auf Papier gedruckt wird.
- Mit der Schaltfläche **Drucker einrichten** wird das Dialogfeld "Druck einrichten" geöffnet, wo Sie Papierformat, Ausrichtung, und weitere Druckeinstellungen für diesen Ausdruck vornehmen können. Siehe auch Befehl "Drucker einrichten".

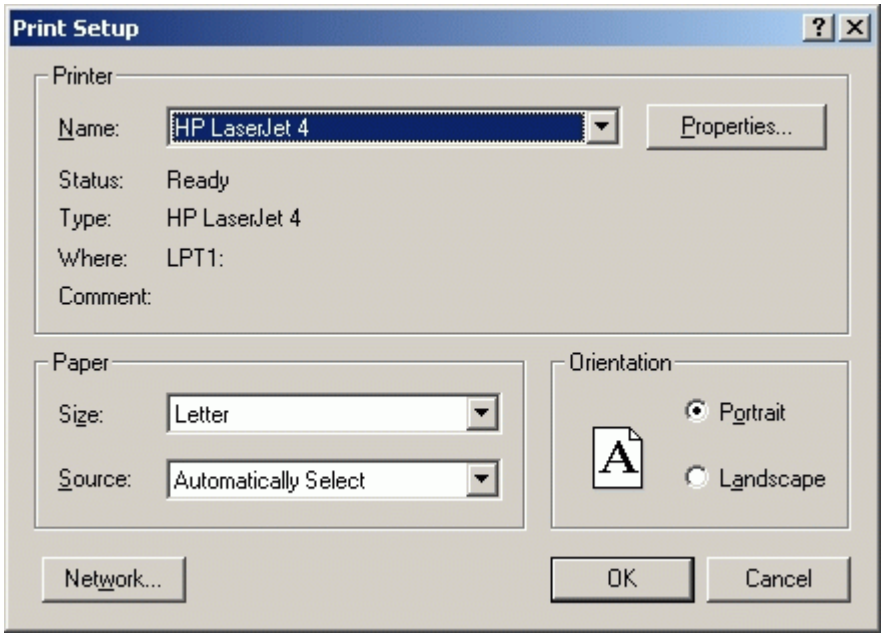
Hinweis: Sie können die Spaltenbreite in der Grid-Ansicht anpassen, um die Druckausgabe zu optimieren.

13.1.14 Druckansicht

Mit dem Befehl **Druckansicht** öffnen Sie das Dialogfeld "Drucken". Klicken Sie auf die Schaltfläche "Vorschau", um eine Druckvorschau des aktiven Dokuments zu sehen.

13.1.15 Druckereinrichtung...

Mit dem Befehl **Druckereinrichtung...** gelangen Sie in das druckerspezifische Dialogfeld, wo Sie Papierformat, Ausrichtung und weitere Einstellungen tätigen können, die für alle zukünftigen Druckaufträge gültig sind.



Das obige Bild zeigt das Dialogfeld, bei dem ein Drucker des Typs HP Laserjet 4 am parallelen Port LPT1 angeschlossen ist.

© 2005 Altova GmbH

13.1.16 Zuletzt verwendete Dateien

Im Menü **Datei** sehen Sie eine Liste der neun zuletzt verwendeten Dateien, wobei die zuletzt verwendete Datei zuoberst angezegit wird. Durch Anklicken einer Datei können Sie diese öffnen. Über die Tastatur können Sie mit der Funktionstaste **ALT+A** das Menü **Datei** aufrufen, und dann die der Datei entsprechende Nummer eintippen.

13.1.17 Beenden

Mit diesem Befehl beenden Sie XMLSpy 2005, nachdem Sie Ihre Arbeit an Ihren Dokumenten abgeschlossen haben. Wenn noch nicht gespeicherte Dokumente offen sind, werden Sie gefragt, ob Sie die Änderungen speichern möchten.
XMLSpy 2005 speichert weiters Programmeinstellungen, sowie Informationen über die zuletzt verwendeten Dateien.

13.2 Menü "Bearbeiten"

In diesem Menü befinden sich Befehle, die beim Editieren mit XMLSpy 2005 verwendet werden.

Zusätzlich zu den Standardbefehlen Rückgängig, Wiederherstellen, Ausschneiden, Kopieren, Einfügen, Löschen, Alles markieren, Suchen, Weitersuchen und Ersetzen, bietet XMLSpy 2005 Sonderbefehle für folgende Funktionen:

- Auswahl in Zwischenablage als XML-Text kopieren,
- Als strukturierten Text kopieren, oder
- XPath Selector für das ausgewählte Objekt in die Zwischenablage kopieren.
- Einfügen und Entfernen von Lesezeichen und Gehe zu Lesezeichen

13.2.1 Rückgängig

Kürzel: **Strg + Z & Alt + Rück**

Dieser Befehl unterstützt unbegrenztes Rückgängigmachen aller Aktionen! Jede durchgeführte Aktion kann in Einzelschritten rückgängig gemacht werden. Der Verlauf der durchgeführten Aktionen wird beim Schließen des Dokuments gespeichert, sodass Aktionen auch zu einem späteren Zeitpunkt zurückverfolgt werden können.

13.2.2 Wiederherstellen

Kürzel: **Strg + Y**

Mit **Wiederherstellen** können Sie zuvor rückgängig gemachte Aktionen wiederherstellen. Sie können den Verlauf des Dokuments sowohl zurück als auch nach vorne verfolgen.

13.2.3 Ausschneiden

Kürzel: **Umschalt + Entf**, oder **Strg + X**

Mit diesem Befehl wird der makierte Text bzw. das markierte Objekt in die Zwischenablage kopiert und aus der aktuellen Position im Dokument gelöscht.

13.2.4 Kopieren

Kürzel: **Strg + C**

Mit diesem Befehl kopieren Sie markierten Text oder Objekte in die Zwischenablage. So können Sie Daten in XMLSpy 2005 duplizieren, oder in eine andere Applikation kopieren.

Hinweis: Es gibt zwei verschiedene Befehle um Elemente in Textform in die Zwischenablage zu kopieren: Als XML-Text kopieren und Als strukturierten Text kopieren. Auf dem Register "Bearbeiten" unter Extras | Optionen können Sie festlegen, welche dieser beiden Operationen bei Auswahl des Befehls "Kopieren" durchgeführt werden soll.

© 2005 Altova GmbH

13.2.5 Einfügen

Kürzel: **Strg + V**

Mit diesem Befehl fügen Sie den Inhalt der Zwischenablage an der aktuellen Cursorposition ein.

13.2.6 Löschen

Tastatur: **Entf**

Mit diesem Befehl wird der markierte Text/das markierte Objekt gelöscht, ohne dass es in die Zwischenablage kopiert wird.

13.2.7 Als XML-Text kopieren

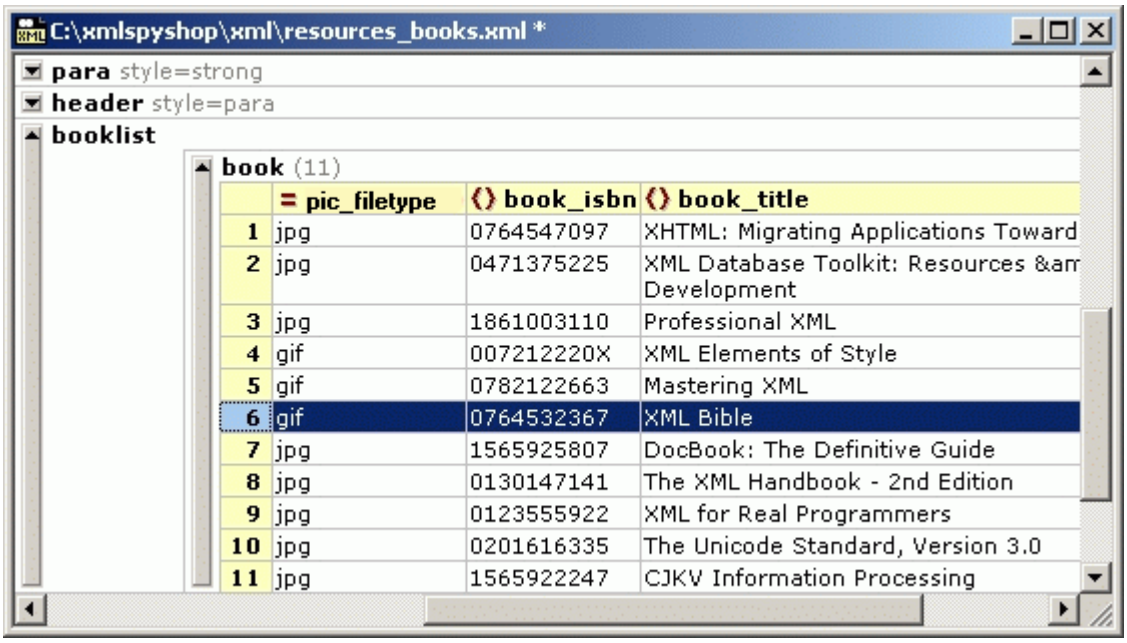
Mit dem Befehl **Als XML-Text kopieren** können Sie auf einfache Weise Daten aus anderen Produkten, die Datenmanipulation auf XML-Ebene erlauben, kopieren. Beim Editieren Ihres Dokuments in der Grid-Ansicht können Sie Elemente in ihrer XML-Textdarstellung in die Zwischenablage kopieren:

```
<row>
    <para align="left">
        <bold>Check the FAQ</bold>
    </para>
    <para>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq30</link_subsection>
            <link_text>XMLSPY 4.0 FAQ</link_text>
        </link>
        <link mode="internal">
            <link_section>support</link_section>
            <link_subsection>faq25</link_subsection>
            <link_text>XMLSPY 3.5 FAQ</link_text>
        </link>
    </para>
</row>
```

Der Befehl formatiert den Text automatisch nach den gültigen Einstellungen für das Speichern der Datei. Diese Einstellungen können im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" im Bereich "Datei speichern" vorgenommen werden.

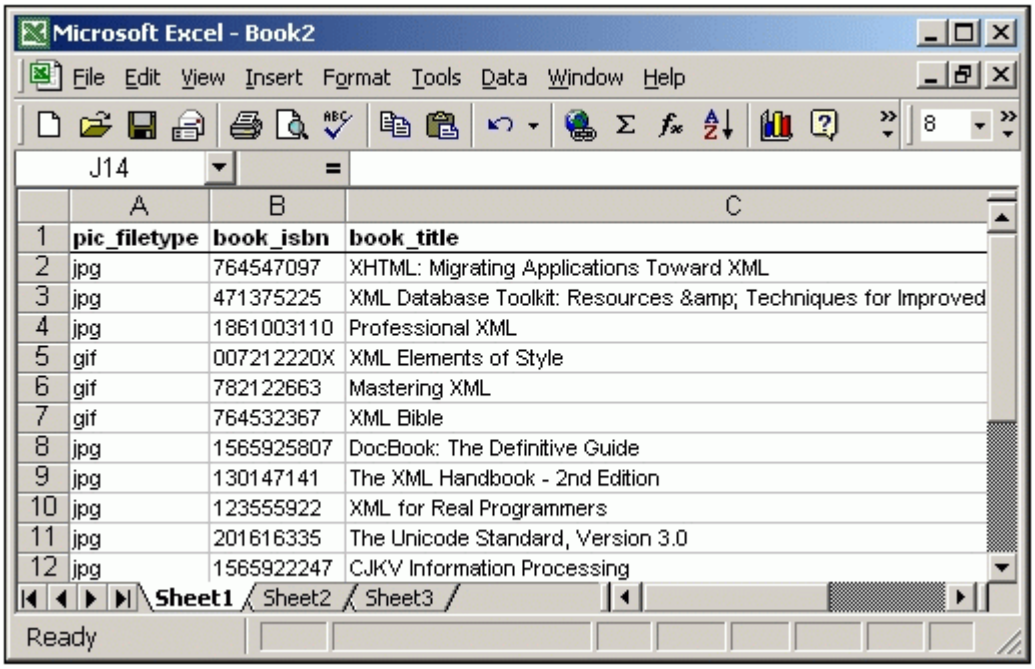
13.2.8 Als strukturierten Text kopieren

Mit diesem Befehl werden Elemente so in die Zwischenablage kopiert, wie sie am Bildschirm angezeigt werden. Dieser Befehl ist beim Kopieren von tabellenförmigen Daten in der Grid-Ansicht sehr nützlich. Die kopierten Daten können in XMLSpy 2005 sowie in Produkten von Drittanbietern verwendet werden. So können Sie z.B. XML-Daten in Spreadsheet-Applikationen wie Microsoft Excel kopieren.



	pic_filetype	book_isbn	book_title
1	jpg	0764547097	XHTML: Migrating Applications Toward
2	jpg	0471375225	XML Database Toolkit: Resources &am Development
3	jpg	1861003110	Professional XML
4	gif	007212220X	XML Elements of Style
5	gif	0782122663	Mastering XML
6	gif	0764532367	XML Bible
7	jpg	1565925807	DocBook: The Definitive Guide
8	jpg	0130147141	The XML Handbook - 2nd Edition
9	jpg	0123555922	XML for Real Programmers
10	jpg	0201616335	The Unicode Standard, Version 3.0
11	jpg	1565922247	CJKV Information Processing

Wenn Sie diese Tabelle nach Excel kopieren, werden Ihre Daten wie folgt angezeigt:



	A	B	C
1	**pic_filetype**	**book_isbn**	**book_title**
2	jpg	764547097	XHTML: Migrating Applications Toward XML
3	jpg	471375225	XML Database Toolkit: Resources & Techniques for Improved
4	jpg	1861003110	Professional XML
5	gif	007212220X	XML Elements of Style
6	gif	782122663	Mastering XML
7	gif	764532367	XML Bible
8	jpg	1565925807	DocBook: The Definitive Guide
9	jpg	130147141	The XML Handbook - 2nd Edition
10	jpg	123555922	XML for Real Programmers
11	jpg	201616335	The Unicode Standard, Version 3.0
12	jpg	1565922247	CJKV Information Processing

Hinweis: Das Ergebnis dieses Befehls hängt davon ab, wie die Daten am Bildschirm angezeigt werden. Wenn das sich das XML-Fragment, das als Beispiel für den Befehl Als XML-Text kopieren verwendet wurde, in der Tabellen-Ansicht der Grid-Ansicht befindet, sieht das Resultat des Kopiervorgangs aus wie folgt:

 © 2005 Altova GmbH

```
row
   para
       align    bold    link
       left     Check the FAQ
           link
              mode       link_section    link_subsection   link_text
              internal   support         faq30             XMLSPY 3.5 FAQ
              internal   support         faq25             XMLSPY 2.5 FAQ
```

In der normalen Grid-Ansicht werden die Daten wie folgt in die Zwischenablage kopiert:

```
row
    para
        align    left
        bold     Check the FAQ
    para
        link
            mode              internal
            link_section      support
            link_subsection   faq30
            link_text         XMLSPY 3.5 FAQ
        link
            mode              internal
            link_section      support
            link_subsection   faq25
            link_text         XMLSPY 2.5 FAQ
```

13.2.9 XPath kopieren

Der Befehl **XPath kopieren** erstellt einen XPath-Ausdruck, der den/die aktuell ausgewählten Node(s) auswählt und den Ausdruck in die Zwischenablage kopiert, sodass Sie ihn dann in ein Dokument (z.B. ein XSLT-Dokument) einfügen können. Alle Ausdrücke beginnen beim Dokument-Root, z.B.:

- `/company/person/last` wenn der Element-Node `last` ausgewählt ist
- `/company/person` wenn ein oder mehrere Element-Nodes `person` ausgewählt sind

Der Befehl "XPath kopieren" ist nur in der Enhanced Grid-Ansicht aktiv.

13.2.10 XML-Text Druckaufbereitung

Mit diesem Befehl wird Ihr XML-Dokument in der Text-Ansicht neu formatiert, sodass Sie eine strukturierte Ansicht Ihres Dokuments erhalten. Jeder Child Node wird um den Betrag eingerückt, den Sie im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register "Datei" unter "Datei speichern" angegeben haben. Diese Funktion kann nur auf wohlgeformte Dokumente angewendet werden.

13.2.11 Alles markieren

Strg + A

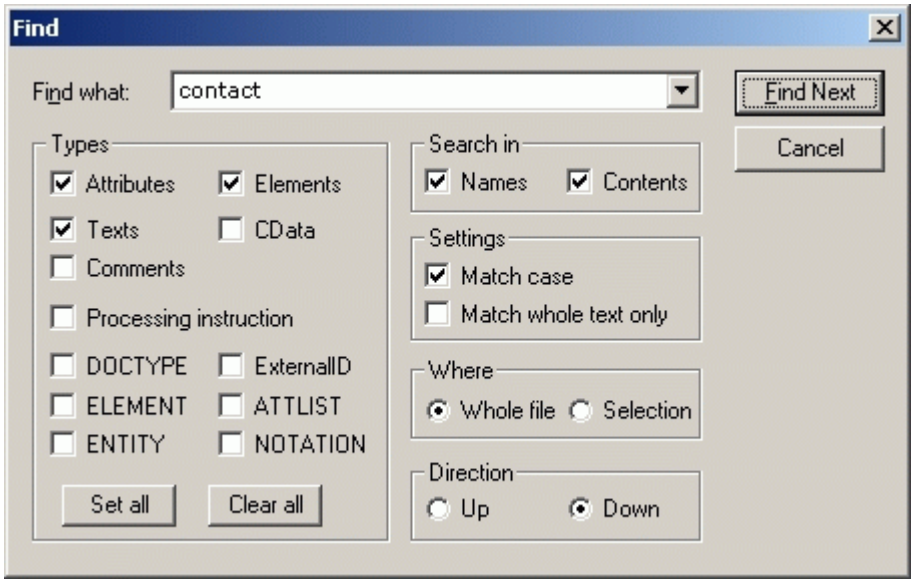
Mit diesem Befehl wird der Inhalt des gesamten Dokuments markiert.

13.2.12 Suchen...

Strg + F

Mit diesem Befehl rufen Sie das Dialogfeld "Suchen" auf, in dem Sie den zu suchenden String sowie weitere Suchoptionen spezifizieren können.
Abhängig von der ausgewählten Ansicht, gibt es im Dialogfeld "Suchen" verschiedene zur Auswahl stehende Optionen.
Um nach Text zu suchen, geben Sie diesen in das Listenfeld "Suchen nach" ein oder wählen Sie einen der letzten 10 Einträge aus der Dropdown-Liste aus und legen Sie anschließend die Suchoptionen fest.

Grid-Ansicht
In der Grid-Ansicht erscheint das folgende Dialogfeld. Die verfügbaren Optionen sind unten beschrieben.

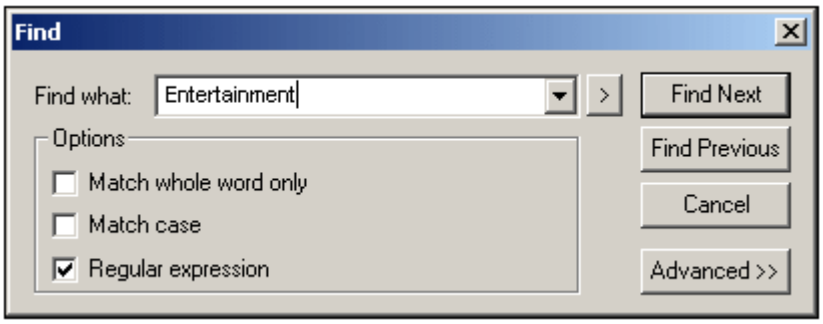


Aktivieren Sie die gewünschten Optionen oder klicken Sie auf ein Optionsfeld.

- Im Bereich "Typ" wird definiert, welche XML-Dokument-Nodes oder -Komponenten in die Suche einbezogen werden sollen. Hierdurch können Sie bestimmte Node-Typen bei der Suche überspringen. Durch Klicken auf die Schaltfläche "Alle aktiv" werden alle Kontrollkästchen im Bereich Typ aktiviert; die Schaltfläche "Zurücksetzen" deaktiviert alle Kontrollkästchen.
- Im Bereich "Suche in Element" können Sie definieren, ob nach Text nur in Node-Namen, dem Inhalt des Node oder in beidem gesucht werden soll.
- Im Bereich "Einstellungen" können Sie angeben, ob die Groß- bzw. Kleinschreibung beachtet werden soll, und ob nur nach ganzen Wörtern gesucht werden soll.
- Im Bereich "Suche in" wird definiert, in welchem Bereich die Suche erfolgen soll.
- Im Bereich "Richtung" definieren Sie, in welche Richtung die Suche erfolgen soll.

© 2005 Altova GmbH

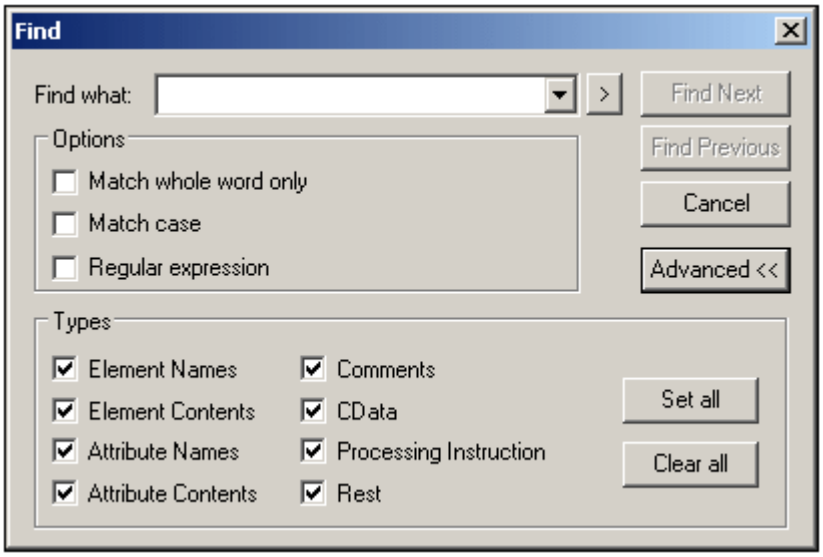
Text-Ansicht
In der Textansicht wird das folgende Dialogfeld angezeigt. Die verfügbaren Optionen sind im Folgenden beschrieben.



Die folgenden Suchoptionen stehen zur Verfügung:

- Ganzes Wort: Bei der Suche werden nur die Wörter im Text gefunden, die exakt mit dem gesuchten Text übereinstimmen.
- GROSS/klein beachten: Bei der Suche wird die Groß- und Kleinschreibung beachtet ("`Address`" ist nicht gleich "`address`").
- Regular expression: Suche nach Text, der durch die Regular Expression definiert wird, die Sie im Textfeld eingegeben haben. Eine Erläuterung zu Regular Expressions finden Sie unter Regular Expressions.

Schaltfläche "Erweitert": Öffnet Gruppe **Typ** (siehe Screenshot unten), wo Sie angeben können, welche XML-Typen durchsucht werden sollen.

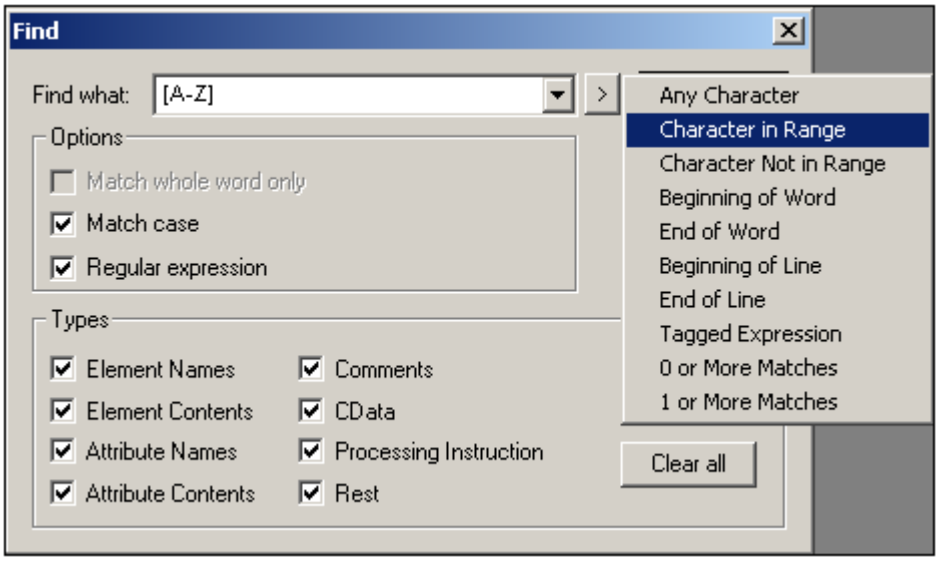


Hinweis:

- Das Suchdialogfeld hat keinen Modus, d.h. es kann geöffnet bleiben, während Sie in der Text-Ansicht arbeiten. Wenn Sie bei geöffnetem Dialogfeld die Eingabetaste drücken, wird es geschlossen. Wenn Text vor dem Öffnen des Dialogfelds markiert war, wird der markierte Text automatisch im Textfeld "Suchen nach:" übernommen.
- Wurde das Suchdialogfeld geschlossen, können Sie die aktuelle Suche wiederholen. Drücken Sie die **F3-**Taste um eine Suche nach unten bzw. **UMSCHALT+F3** um eine Suche nach oben durchzuführen.
- Durch Klicken auf die Erweiterungsschaltfläche rechts neben dem "Suchen nach:"-

Listenfeld wird ein zweites Fenster geöffnet, über das Sie Regular Expressions eingeben können.

Regular Expressions:
Sie können Ihre Suchkriterien mit Hilfe von Regular Expressions näher definieren. Für die Erstellung der Regular Expression steht eine Popup-Liste zur Verfügung, die Sie über die >-Schaltfläche rechts vom Suchbegriff-Eingabefeld aufrufen können.



Wenn Sie auf die Beschreibung des gewünschten Suchausdrucks klicken, wird die entsprechende Expression-Syntax im Eingabefeld übernommen. Im Folgenden sehen Sie eine Liste von Syntax-Zeichen, die für Regular Expressions verwendet werden.

© 2005 Altova GmbH

.	Steht für jedes beliebige Zeichen. Dies ist ein Platzhalter für ein einzelnes Zeichen.
\(	Markiert den Beginn einer Region, die gesucht werden soll.
\)	Markiert das Ende einer zu suchenden Region.
\n	n steht für 1 bis 9 und bezieht sich auf die erste bis neunte zu ersetzende Suchentsprechung. Lautet der Suchstring beispielsweise Fred\([1-9]\)XXX und der Ersetzungsstring Sam\1YYY, so wird Fred2XXX durch Sam2YYY ersetzt.
\<	Steht für den Beginn eines Worts.
\>	Steht für das Ende eines Worts.
\x	Damit können Sie ein Zeichen x verwenden, das sonst eine spezielle Bedeutung hätte. So würde z.B. \[als [und nicht als Beginn einer Zeichenmenge interpretiert werden.
[...]	Kennzeichnet eine Zeichenmenge, [abc] z.B. steht für jedes der Zeichen a, b oder c. Sie können auch Bereiche angeben, z.B. [a-z] für alle klein geschriebenen Zeichen.
[^...]	Die invertierte Zeichenmenge. [^A-Za-z] z.B. steht für jedes Zeichen mit Ausnahme alphabetischer Zeichen.
^	Steht für den Zeilenanfang (es sei denn dieses Zeichen wird innerhalb einer Menge verwendet, siehe oben).
$	Steht für das Zeilenende. Beispiel: A+$ findet ein oder mehrere A's am Ende der Zeile.
*	Steht für 0 oder öfter. Mit Sa*m werden z.B. Sm, Sam, Saam, Saaam usw. gefunden.
+	Steht für 1 oder öfter. Mit Sa+m werden z.B. Sam, Saam, Saaam usw. gefunden.

.

13.2.13 Weitersuchen

F3

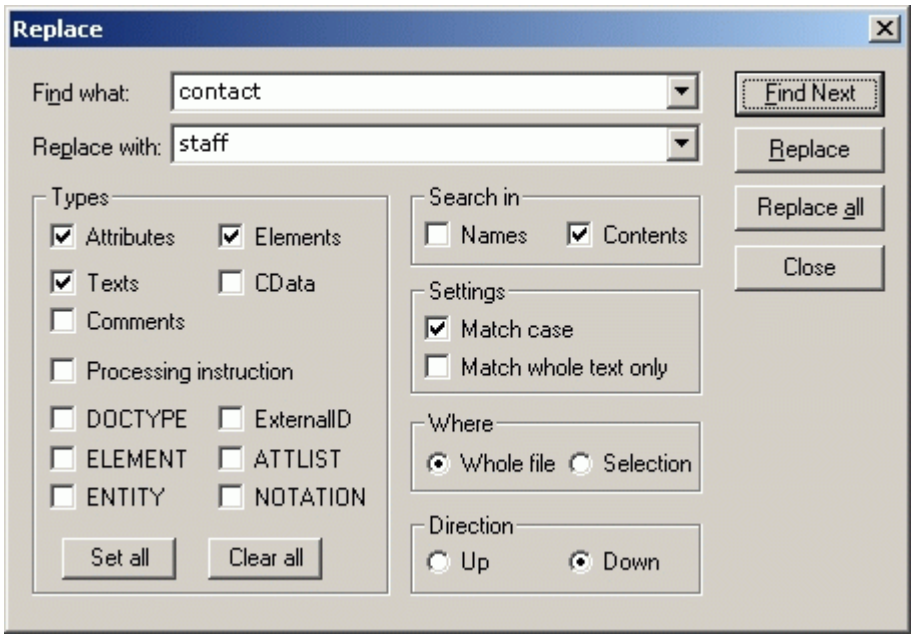
Mit **Weitersuchen** wird weiter nach dem zuvor eingegebenen Begriff gesucht.

13.2.14 Ersetzen...

Strg + H

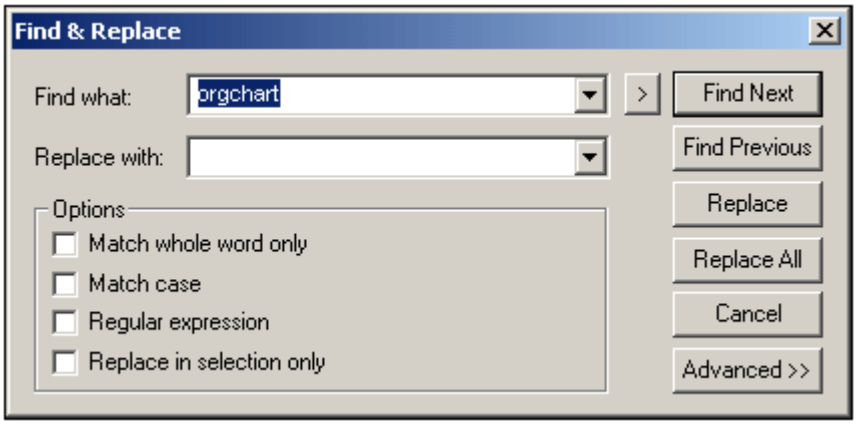
Mit dem Befehl **Ersetzen** können Sie einen Textstring durch einen anderen ersetzen. Der Befehl verfügt über dieselben Optionen wie der Befehl Suchen... und ermöglicht es Ihnen, jeden beliebigen Text durch einen anderen Text Ihrer Wahl zu ersetzen.
Abhängig von der ausgewählten Ansicht, gibt es im Dialogfeld "Ersetzen" verschiedene zur Auswahl stehende Optionen. Sie können diese Funktion Schritt für Schritt verwenden, oder alle Begriffe auf einmal ersetzen, indem Sie auf die Schaltfläche "Alle ersetzen" klicken.

Grid-Ansicht
In der unten gezeigten Abbildung sehen Sie die verschiedenen Suchoptionen.



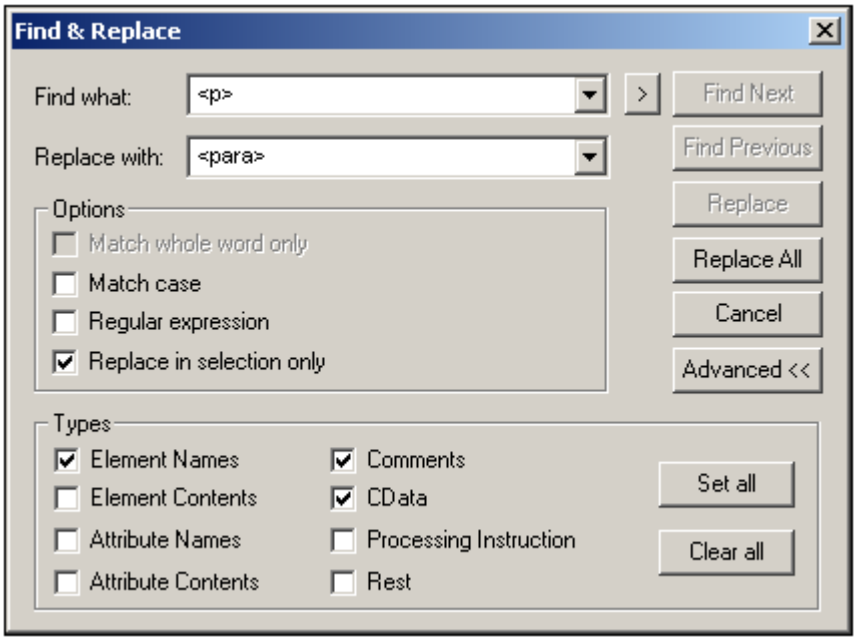
Diese Optionen sind im Abschnitt Suchen... beschrieben.

Text-Ansicht
Wenn Sie in der Text-Ansicht den Befehl "Ersetzen" wählen, wird das unten gezeigte Dialogfeld "Suchen & Ersetzen" angezeigt. Die Optionen sind dieselben wie im Dialogfeld Suchen..... Bei Auswahl der Option "Nur im markierten Bereich ersetzen" werden die Ersetzungen nur im markierten Dokumentbereich vorgenommen.



Wenn Sie auf die Schaltfläche "Erweitert" klicken, werden die Typ-Optionen angezeigt (nähere Informationen dazu siehe Suchen...).

© 2005 Altova GmbH



Hinweis: Bei Auswahl des Befehls "Alle ersetzen" wird jede einzelne Ersetzung als Einzelvorgang aufgezeichnet, sodass Sie diesen Vorgang Schritt für Schritt rückgängig machen können.

13.2.15 Lesezeichen einfügen/löschen

Strg+F2

Mit dem Befehl **Lesezeichen einfügen/löschen** können Sie ein Lesezeichen an der aktuellen Cursorposition einfügen, bzw. ein vorhandenes Lesezeichen entfernen, wenn sich der Cursor in einer Zeile befindet, in der zuvor ein Lesezeichen eingefügt wurde. Dieser Befehl steht nur in der Textansicht zur Verfügung.

Mit Lesezeichen markierte Zeilen können auf zwei Arten angezeigt werden:

- Wurde die Lesezeichenleiste aktiviert, wird eine blaue Ellipse links vom Text in der Lesezeichenleiste angezeigt.
- Ist die Lesezeichenleiste nicht aktiviert, so wird die gesamte Zeile, in der sich der Cursor befindet markiert.

Mit der **F2**-Taste können Sie der Reihe nach alle Lesezeichen in der Textansicht anzeigen.

13.2.16 Alle Lesezeichen löschen

Strg+Umschalt+F2

Mit diesem Befehl löschen Sie alle derzeit in der Textansicht definierten Lesezeichen. Beachten Sie, dass dieser Befehl nicht rückgängig gemacht werden kann. Dieser Befehl steht nur in der Textansicht zur Verfügung.

13.2.17 Zum nächsten Lesezeichen

F2

Setzt den Cursor an den Beginn der nächsten mit einem Lesezeichen versehenen Zeile. Dieser Befehl ist nur in der Textansicht verfügbar.

13.2.18 Zum vorhergehenden Lesezeichen

Umschalt+F2

Setzt den Textcursor an den Beginn der vorhergehenden mit einem Lesezeichen versehenen Zeile. Dieser Befehl ist nur in der Textansicht verfügbar.

© 2005 Altova GmbH

13.3 Menü "Projekt"

XMLSpy 2005 verwendet die bekannte Baumstruktur, um Dateien und URLs in einem XML-Projekt zu gliedern. Dateien und URLs mit gleicher Erweiterung können in Ordnern gesammelt werden, wodurch eine einfache Strukturierung und Batchmanipulation gegeben ist.

Bitte beachten Sie:
Die meisten projektbezogenen Befehle sind auch mittels Rechtsklick auf das gewünschte Objekt über das Kontextmenü verfügbar.

Absolute und relative Pfade
Jedes Projekt ist als Projektdatei gespeichert, die mit der Erweiterung **.spp** versehen ist. Bei diesen Dateien handelt es sich eigentlich um XML-Dokumente, die Sie wie jede reguläre XML-Datei editieren können. In der Projektdatei werden für Dateien/Ordner auf derselben oder einer höheren Ebene absolute Pfade verwendet, während für Dateien/Ordner im aktuellen Ordner oder einem Unterordner relative Pfade definiert werden. Wenn Ihre Verzeichnisstruktur z.B. in etwa so aussieht:

```
|-Folder1
|      |
|      |-Folder2
|             |
|             |-Folder3
|                    |
|                    |-Folder4
```

...und Ihre **.spp-**Datei in `Folder3` gespeichert ist, so werden Referenzen auf Dateien in `Folder1` und `Folder2` folgendermaßen angegeben:

```
c:\Folder1\NameOfFile.ext
c:\Folder1\Folder2\NameOfFile.ext
```

Referenzen auf Dateien in `Folder3` und `Folder4` würden in etwa folgendermaßen aussehen:

```
.\NameOfFile.ext
.\Folder4\NameOfFile.ext
```

Wenn Sie sicherstellen möchten, dass alle Pfade relativ angegeben werden, speichern Sie die `.spp`-Dateien im Root-Verzeichnis Ihres Rechners.

Projekte und Versionskontrolle
Wenn Sie vorhaben, ein XMLSpy 2005-Projekt zu einem Versionskontroll-Repository hinzuzufügen, stellen Sie bitte sicher, dass die Projektdateien in der hierarchischen Dateisystemstruktur in einem Ordner abgelegt werden, von dem aus Dateien nur aus diesem untergeordneten Ordnern hinzugefügt werden können (Nehmen Sie das Root-Verzeichnis als oberste Ebene der Verzeichnisstruktur)

In anderen Worten stellt das Verzeichnis, in dem sich die **Projektdatei** befindet, innerhalb des Versionskontroll-Repository im Grunde das **Root-Verzeichnis** des Projekts dar. Dateien, die dem Projekt aus einem Ordner hinzugefügt werden, der sich oberhalb dieses Root-Verzeichnisses befindet, werden dem XMLSpy 2005-Projekt zwar hinzugefügt, doch ist ihr Speicherort im Repository - falls überhaupt zulässig - manchmal ein unerwarteter.

Nehmen wir als Beispiel die oben gezeigte Verzeichnisstruktur:
Wenn eine Projektdatei in `Folder3` gespeichert wird und einer Versionskontrolle unterliegt:

- so werden Dateien, die zu `Folder1` hinzugefügt werden, unter Umständen nicht von der Versionskontrolle berücksichtigt,
- werden Dateien, die zu `Folder2` hinzugefügt werden, dem Root-Verzeichnis des

Repository anstelle des Projektordners hinzugefügt, jedoch von der Versionskontrolle berücksichtigt,
- während die in `Folder3` und `Folder4` gespeicherten Dateien korrekt verarbeitet werden und auch der Versionskontrolle unterliegen.

13.3.1 Neu

Mit diesem Befehl erstellen Sie ein **neues** Projekt in XMLSpy 2005.
Wenn Sie gerade an einem anderen Projekt arbeiten, werden Sie gefragt, ob Sie alle Dokumente schließen wollen, die zu dem aktuellen Projekt gehören.

13.3.2 Öffnen...

Diese Funktion **öffnet** ein bereits vorhandenes Projekt inXMLSpy 2005.
Falls Sie zuvor an einem anderen Projekt gearbeitet haben, wird dieses Projekt zuerst geschlossen.

13.3.3 Neu laden

Mit dieser Funktion wird das aktuelle Projekt von der Festplatte **neu geladen**.
Falls Sie in einer Multiuser-Umgebung arbeiten, kann es oftmals nötig sein, ein Projekt, an dem mehrere Personen gleichzeitig arbeiten, neu zu laden, um deren Änderungen anzusehen.

Hinweis: Projektdateien (.spp) sind XML-Dokumente, die Sie wie jede andere XML Datei bearbeiten können.

13.3.4 Schließen

Mit diesem Befehl schließen Sie das **aktuelle** Projekt. Wurde das Projekt geändert, werden Sie vor dem Schließen gefragt, ob die Änderungen gespeichert werden sollen.
Wenn ein Projekt oder eine XML-Datei geändert wurde, wird der Name der Datei in der Titelleiste des Projekts mit einem Stern "*" versehen.

13.3.5 Speichern

Mit diesem Befehl wird das aktuelle Projekt **gespeichert**. Sie können ein Projekt auch speichern, indem Sie das Projektfenster zum aktiven machen und auf das Symbol klicken.

13.3.6 Versionskontrolle

XMLSpy 2005 unterstützt Microsoft Source-Safe und andere kompatible Speicherschnittstellen.

© 2005 Altova GmbH

Open Source Versionskontrollsysteme:

- Concurrent Versions System (CVS).
  Um XMLSpy 2005 mit CVS verbinden zu können, müssen Sie die Software Jalindi Igloo herunterladen. Sie finden diese Softwarekomponente entweder auf der Altova -Komponenten-Website im Abschnitt Komponenten oder auf der Website von Jalindi.com .

Kommerzielle Versionskontrollsysteme:

- Microsoft Visual Source-Safe 6.0c
- PVCS Version Manager 6.7
- StarTeam 5.1

Ein Versionskontrollprojekt ist allerdings nicht mit einem Projekt in XMLSpy 2005 zu vergleichen. Versionskontrollprojekte sind verzeichnisbezogen, wobei XMLSpy 2005 Projekte logische Konstrukte ohne direkte Verzeichnisabhängigkeit sind.

Im folgenden Beispiel wird ein Text verwendet, der Microsoft Source-Safe als Versionskontrolle benutzt.

Bitte beachten Sie:
Microsoft hat einen Registryeintrag definiert, bei dem alle SCC-kompatiblen Programme sich selbst registrieren. XMLSpy 2005 liest nur diesen Eintrag.

HKEY_LOCAL_MACHINE\SOFTWARE\SourceCodeControlProvider\InstalledSCCProviders

Delta V
Erweitert das Protokoll WebDAV, wobei WebDAV eine Erweiterung des Protokolls HTTP 1.1 ist. XMLSpy 2005 unterstützt Delta V auf Projekt,- Verzeichnis-, Ordner- und Dateiebene über das Kontextmenü, das mit Rechtsklick auf das gewünschte Objekt aufgerufen wird. Das DeltaV Protokoll wird von XMLSpy 2005 unterstützt, wird aber nur verwendet, wenn der Server, auf dem die Dateien unter Versionskontrolle stehen, dies zulässt.

Beim Zugriff auf Dateien, die auf einem Server abgelegt sind, der DeltaV unterstützt, werden automatisch die Funktionen Einchecken/Auschecken/Auschecken rückgängig aktiviert.

Delta V Befehle werden durch Rechtsklick auf folgende Objekte aufgerufen:

- Dateiregister einer offenen Datei im Hauptfenster
- Datei im Fenster "URL öffnen"
- Datei in einem externen Webverzeichnis eines Projekts

Delta V ist der **Versionierungsteil** des WebDAV Standards, und unterstützt:

- Auschecken
- Auschecken rückgängig
- Einchecken (inkl. Kommentare)

WebDAV Merkmale:

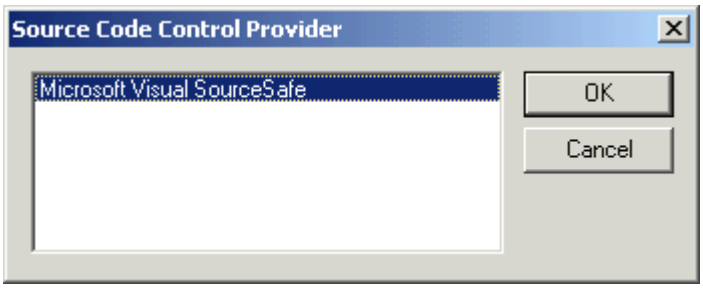
- Sicherung gegen Überschreiben: Lock und Unlock
- Eigenschaften: Auflisten, hinzufügen, löschen
- Namespace Aktionen: Verschieben und Kopieren
- Sammlungen: Hierarchieoperationen und mkcol.

Weitere Informationen über DeltaV finden Sie unter www.webdav.org/deltav.

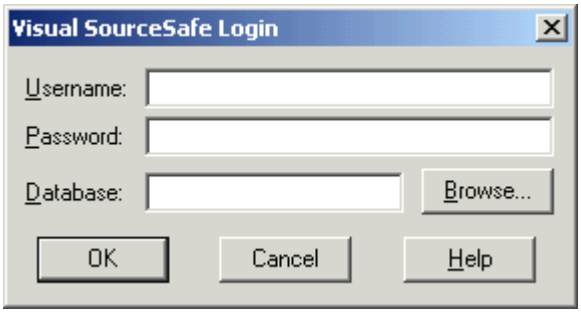
Öffnen

Mit der Funktion "Öffnen" wird ein bereits vorhandenes Versionskontrollprojekt lokal wiedergegeben (oder ein neues Projekt erstellt). Diese Funktion kann nur ausgeführt werden, wenn ein XMLSpy 2005 Projekt zuvor in die Versionskontrolle mittels der Menüoption Zu Versionskontrolle hinzufügen übernommen wurde.

1. Wählen Sie eine Versionskontrolle, die auf Ihrem PC installiert ist, und bestätigen Sie mit der Schaltfläche OK.

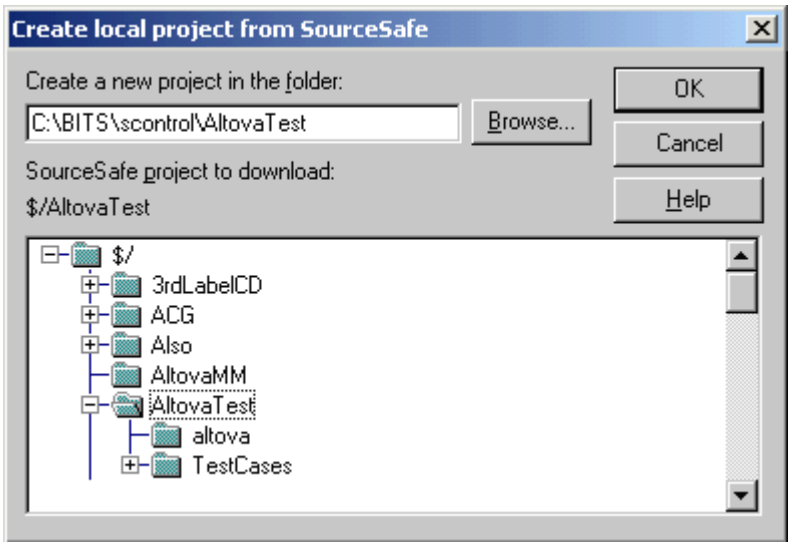


2. Geben Sie Ihre Login-Daten im Versionskontrollfenster ein, und bestätigen Sie mit OK.

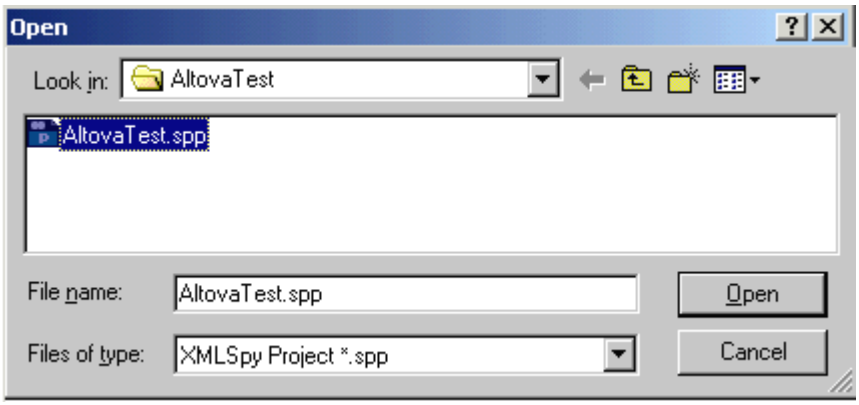


3. Geben Sie an, in welchem Verzeichnis das lokale Projekt abgelegt werden soll.
4. Wählen Sie das herunterzuladende Projekt.
   Falls das gewählte Verzeichnis noch nicht existiert, werden Sie aufgefordert, es zu erstellen. Klicken Sie auf "Ja" zum Erstellen des neuen Ordners.
5. Klicken Sie auf die Schaltfläche OK um das Versionskontrollprojekt herunterzuladen.

© 2005 Altova GmbH

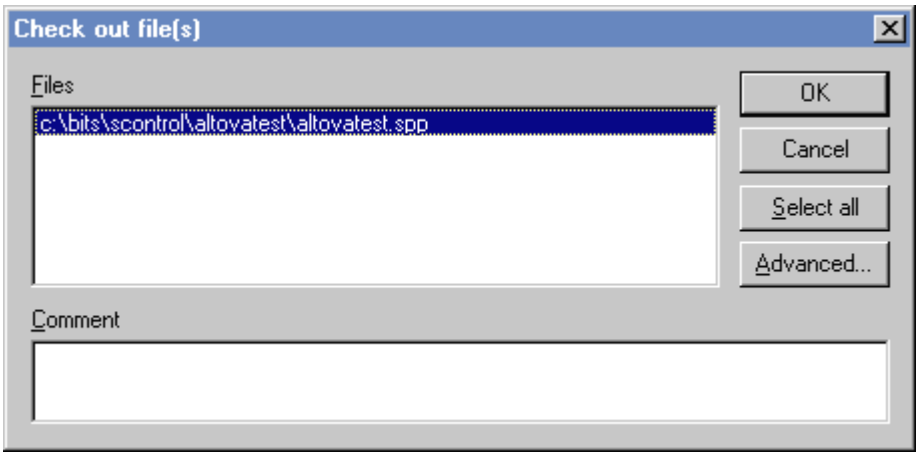


6. Das Dialogfeld "Öffnen" wird angezeigt. Hier können Sie die XMLSpy 2005 Projektdatei (*.spp) im lokalen Verzeichnis suchen. Klicken Sie auf die zu öffnende Projektdatei.



Falls sich das Projekt bereits unter Versionskontrolle befindet, erhalten Sie eine Meldung, dass das Projekt unter Versionskontrolle steht. Sie können nun entscheiden, ob Sie das Projekt auschecken möchten.

7. Klicken Sie auf die Schaltfläche OK um die Versionskontrolle zu aktivieren. Das Dialogfeld Auschecken wird geöffnet. Nun können Sie Ihre *.spp Datei auschecken.
8. Im Textfeld **Kommentare** können Sie Kommentare zu Dateien eingeben. Klicken Sie auf OK um die XMLSpy 2005 Projektdatei auszuchecken. Je nach Versionskontrollsystem enthält das Dialogfeld unter Umständen weitere Befehlsschaltflächen. Mit der Schaltfläche **Erweitert** öffnen Sie das Dialogfeld "Erweiterte Checkout-Funktionen".



Die *.spp Datei ist nun ausgecheckt, und alle XML Dateien, die im XMLSpy 2005 Projekt enthalten sind, werden in das lokale Verzeichnis kopiert.

Resultat in XMLSpy 2005:
Das AltovaTest Verzeichnis wurde lokal erstellt. Alle Dateien des XMLSpy 2005 Projekts werden nun im Projektfenster angezeigt. Sie können nun mit diesen Dateien arbeiten.

Symbole der Versionskontrolle:

Das **rote Kontrollhäkchen** zeigt an, dass diese Datei **ausgecheckt** ist, z.B.ist die Projektdatei "AltovaTest" ausgecheckt.
Das Sternchen am Ende des Dateinamens bedeutet, dass diese Datei verändert wurde. Beim Schließen werden Sie aufgefordert, die Datei zu speichern.

Das Vorhängeschloss-Symbol oben rechts im XML Symbol zeigt an, dass diese Datei **unter Versionskontrolle steht**, aber derzeit nicht ausgecheckt ist.

Dieses URL-Dokument ist Teil des AltovaTest Projektes, steht aber **nicht unter Versionskontrolle**. Dokumente, die über eine URL geöffnet wurden, können nicht unter Versionskontrolle gestellt werden.

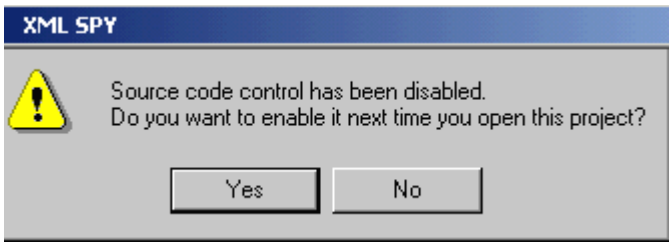
Versionskontrolle aktivieren

Diese Funktion erlaubt es, die Versionskontrolle für ein Projekt zu aktivieren oder deaktivieren.

So deaktivieren Sie die Versionskontrolle für ein Projekt:

1. Klicken Sie auf die Projektdatei oder eine beliebige Datei im Projektfenster.
2. Wählen Sie die Menüoption **Projekt | Versionkontrolle** und deaktivieren Sie das Kontrollkästchen **Versionskontrolle aktivieren**.

 © 2005 Altova GmbH



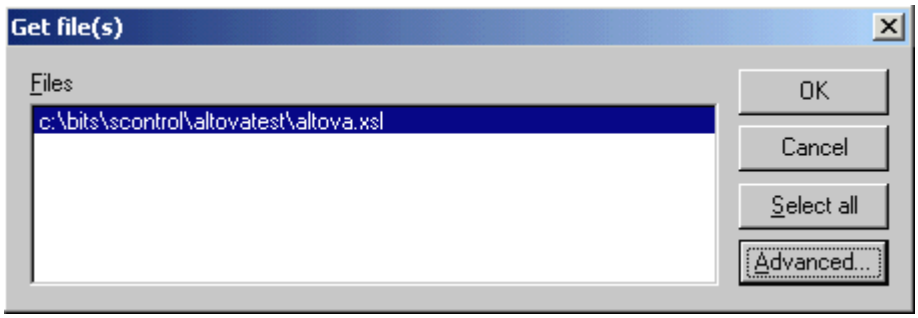
Um die Versionskontrolle **provisorisch** zu deaktivieren, wählen Sie Ja.

Um die Versionskontrolle für dieses Projekt **permanent** zu deaktivieren, wählen Sie Nein.

Aktuellste Version holen

Mit diesem Befehl wird die aktuellste Version der ausgewählten Datei geholt.

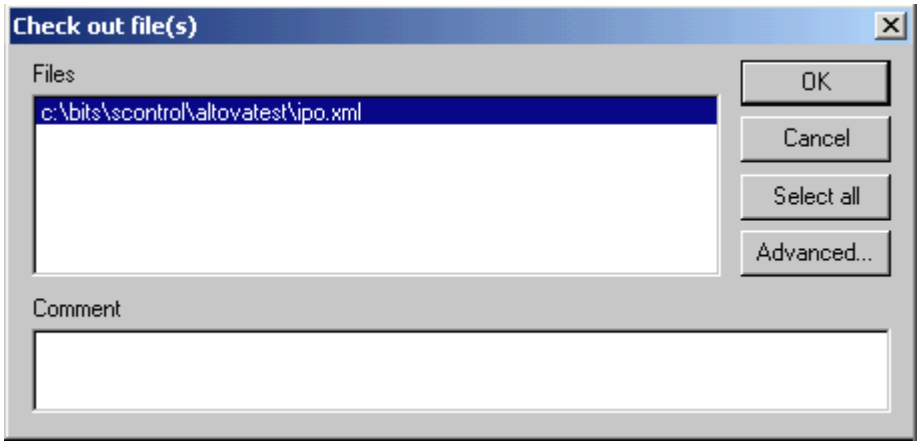
Diese Datei ist eine Kopie der Datei unter Versionskontrolle und wird in der lokalen Arbeitsmappe Ihres Computers abgelegt.



Auschecken

Mit diesem Befehl checken Sie die aktuellste Version der ausgewählten Datei(en) aus. Die Dateien werden für andere Benutzer als "ausgecheckt" markiert.

Kürzel: Rechtsklicken Sie auf ein Objekt im Projektfenster, und wählen Sie "Auschecken" im Kontextmenü.



Folgende Objekte können ausgecheckt werden:

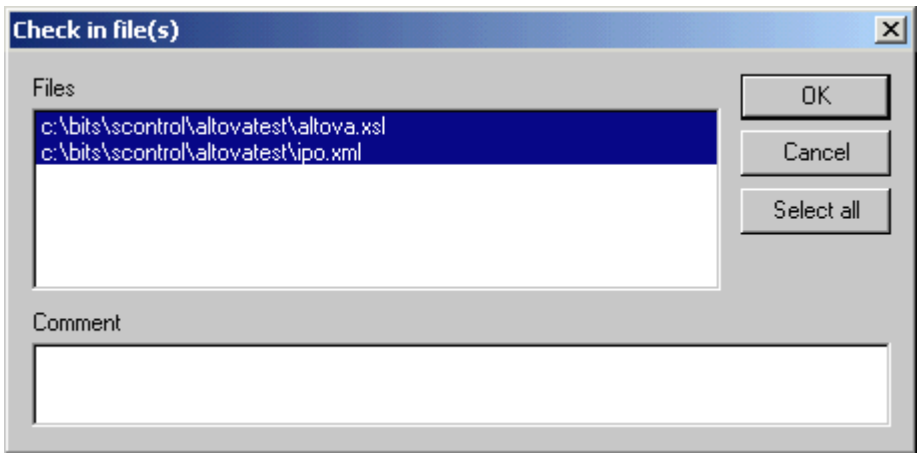
- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy 2005 Projektordner - klicken Sie auf den Ordner (oder mehrere Ordner mittels Strg + Klick)
- XMLSpy 2005 Projekt - klicken Sie auf das Symbol der Projektdatei.

Das rote Kontrollhäkchen zeigt an, dass die Datei ausgecheckt ist.

Einchecken

Mit diesem Befehl checken Sie zuvor ausgecheckte Dateien wieder ein (z.B. lokal aktualisierte Dateien), und stellen die Dateien zurück in das Versionskontrollprojekt.

Kürzel: Rechtsklicken Sie auf ein ausgechecktes Objekt, und wählen Sie "Einchecken" im Kontextmenü.



Folgende Objekte können eingecheckt werden:

- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy 2005 Projektordner - klicken Sie auf den Ordner (odere mehrere Ordner mittels Strg + Klick)
- Ein komplettes XMLSpy 2005 Projekt - klicken Sie auf das Symbol der Projektdatei.

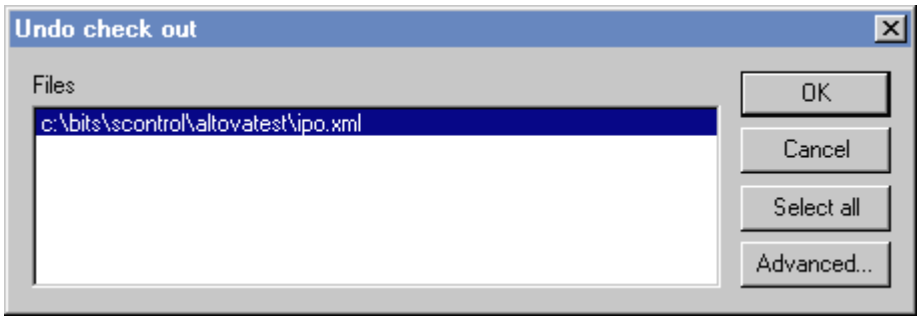
Das Vorhängeschloss-Symbol zeigt an, dass sich die Datei **unter Versionskontrolle** befindet, aber derzeit nicht ausgecheckt ist.

Auschecken rückgängig...

Mit diesem Befehl werden Änderungen in zuvor ausgecheckten Dateien **verworfen**. Lokal aktualisierte Dateien können gegen ältere Dateien im Versionskontrollprojekt ausgetauscht werden.

Kürzel: Rechtsklicken Sie auf ein Objekt im Projektfenster, und wählen Sie "Auschecken rückgängig" im Kontextmenü.

© 2005 Altova GmbH

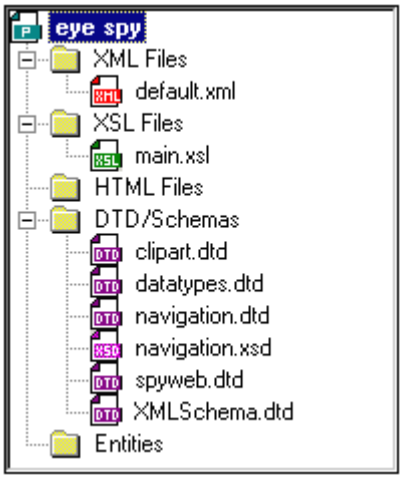


Auschecken rückgängig kann auf folgende Objekte angewendet werden.

- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy 2005 Projektordner - klicken Sie auf den Ordner (odere mehrere Ordner mittels Strg + Klick)
- XMLSpy 2005 Projekt - klicken Sie auf das Symbol der Projektdatei.

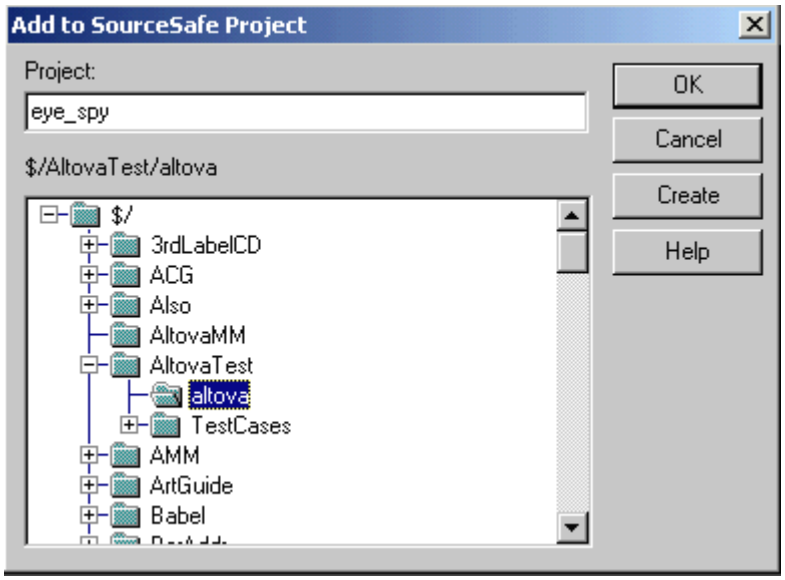
Zu Versionskontrolle hinzufügen

Dieser Befehl fügt ein XMLSpy 2005 Projekt in Ihre Versionskontrolle ein.

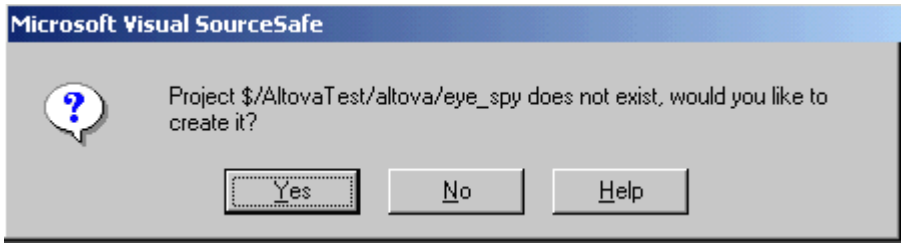


So fügen Sie ein XML Projekt einer Versionskontrolle hinzu:

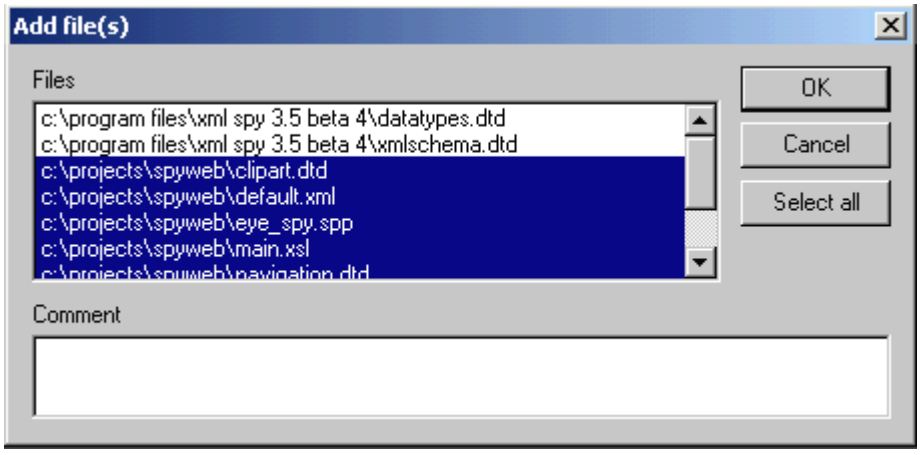
1. Klicken Sie auf das XMLSpy 2005 Projektsymbol, und wählen Sie die Menüoption **Projekt | Versionskontrolle | Zu Versionskontrolle hinzufügen.**
2. Wählen Sie Ihr Versionkontrollprodukt aus der Liste.
3. Geben Sie die benötigten Login-Daten ein.
4. Wählen Sie das Versionskontrollprojekt (Verzeichnis), in das Ihr XML Projekt eingefügt werden soll, und bestätigen sie mit OK.



Falls das Projekt nicht existiert, öffnet sich folgendes Dialogfeld:



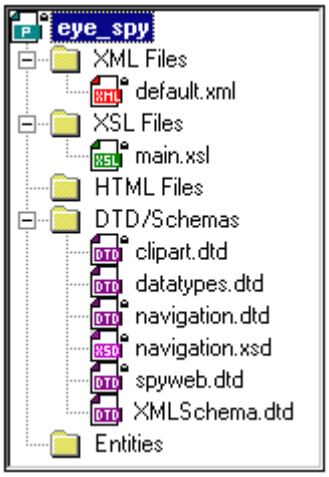
5. Wählen Sie Ja um ein neues Projekt zu erstellen. Mit Nein brechen Sie diesen Vorgang ab.



6. Wählen Sie die XMLSpy 2005 Projektdateien, die Sie unter Versionskontrolle stellen möchten aus, und bestätigen Sie mit OK.

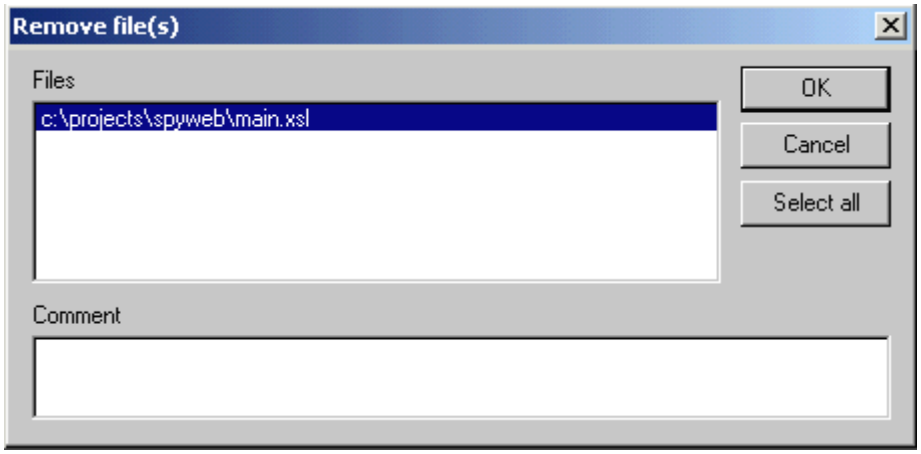
Diese Dateien werden nun in die Versionskontrolle eingefügt, und befinden sich unter Versionskontrolle. Die Dateien sind durch das Schloss-Symbol gekennzeichnet. Nicht transferierte Dateien haben kein Schloss-Symbol (in der unteren Abbildung: datatypes.dtd und XMLschema.dtd).

© 2005 Altova GmbH



Von Versionskontrolle ausgliedern

Mit diesem Befehl **löschen** Sie zuvor hinzugefügte Dateien aus dem Versionskontrollprodukt.



Folgende Objekte könne aus der Versionskontrolle ausgegliedert werden:

- Einzelne Dateien - klicken Sie auf die Datei (oder mehrere Dateien mittels Strg + Klick)
- XMLSpy 2005 Projektordner - klicken Sie auf den Ordner (oder mehrere Ordner mittels Strg + Klick)
- XMLSpy 2005 Projekt - klicken Sie auf das Symbol der Projektdatei.

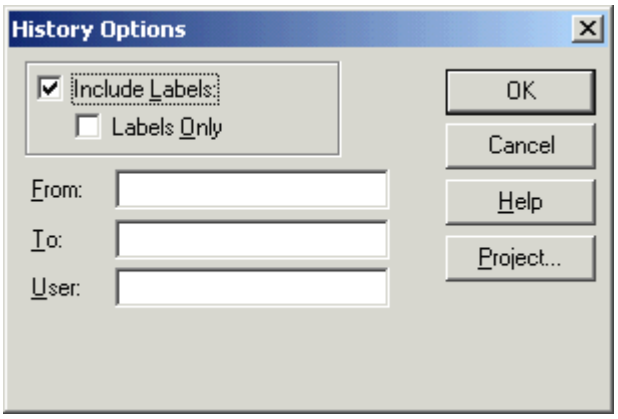
Verlauf anzeigen

Mit diesem Befehl zeigen Sie den Verlauf einer bestimmten Datei an. Der Verlauf kann jeweils nur für eine Datei angezeigt werden.

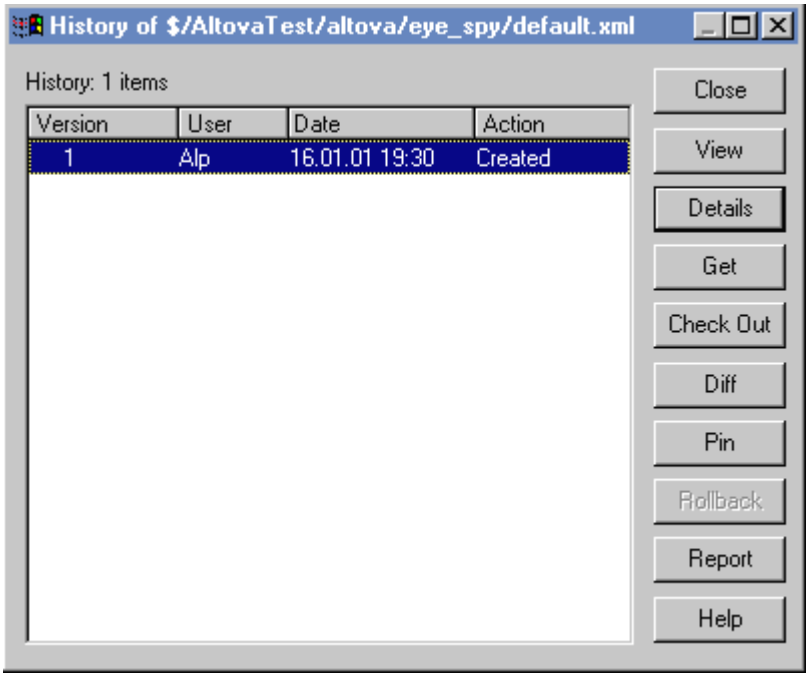
So zeigen Sie den Verlauf einer Datei an:

1. Klicken Sie auf die Datei im Projektfenster, deren Verlauf Sie ansehen möchten.
2. Wählen Sie die Menüoption **Projekt | Versionskontrolle | Verlauf anzeigen**.

   Im folgenden Dialogfeld können Sie weitere Informationen eingeben (in diesem Beispiel handelt es sich um ein Dialogfeld von Microsoft Source-Safe).



3. Wählen Sie den gewünschten Eintrag und bestätigen Sie mit OK.



Unterschiede anzeigen

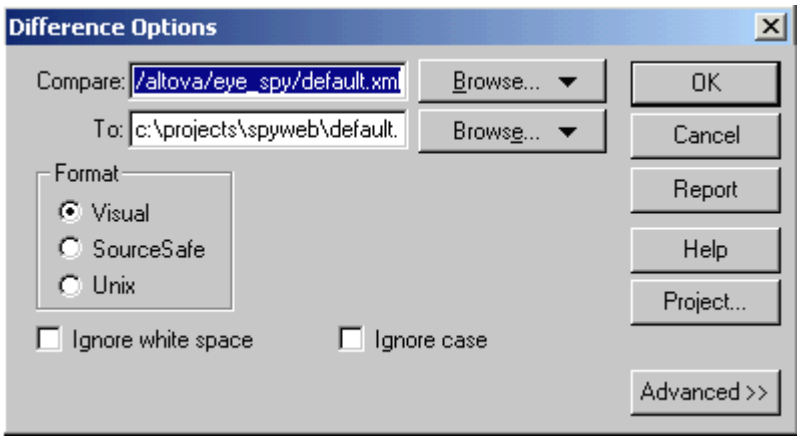
Dieser Befehl ermöglicht die Anzeige von Unterschieden zwischen der in der Versionskontroll-Ablage befindlichen und der gleichnamigen Datei, die Sie ausgecheckt haben.
Dieser Befehl kann jeweils nur auf eine einzelne Datei angewendet werden.

So zeigen Sie die Unterschiede zweier Dateien an:

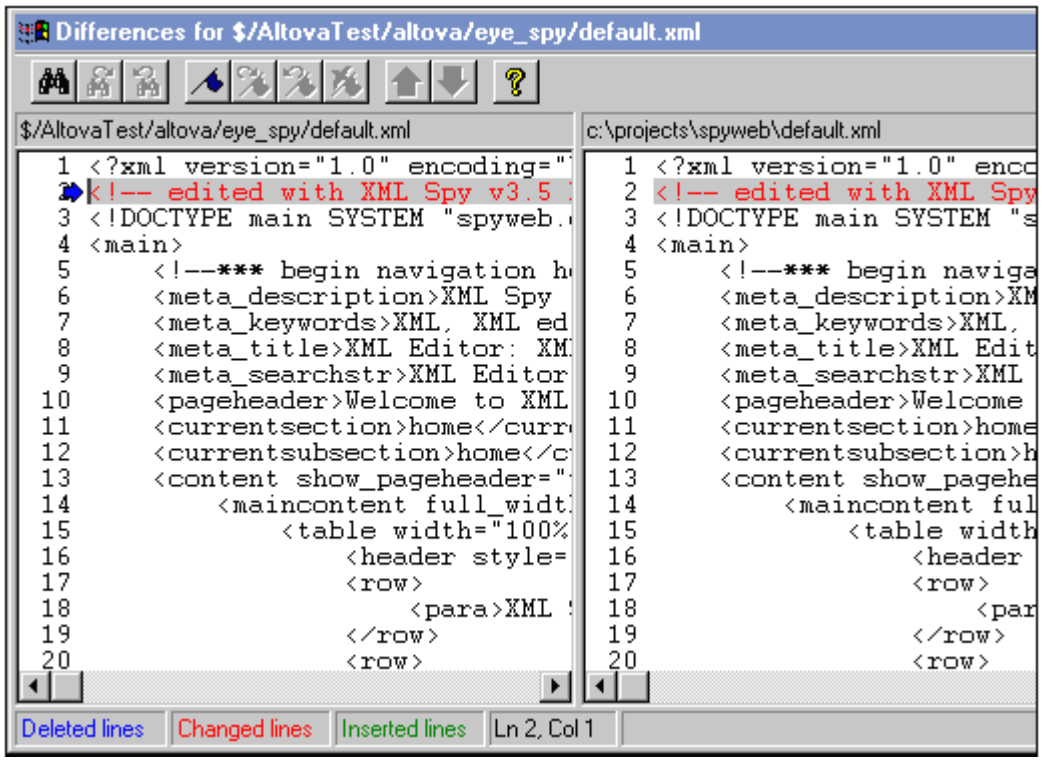
Nachdem Sie eine Datei aus dem Projekt ausgecheckt haben:

1. Klicken Sie auf die Datei im Projektfenster.
2. Wählen Sie die Menüoption **Projekt | Versionskontrolle | Unterschiede anzeigen.** Ein Dialogfeld kann an dieser Stelle erscheinen, in dem Sie weitere Informationen eingeben können.

© 2005 Altova GmbH



3. Wählen Sie die gewünschten Einträge und bestätigen Sie mit OK.

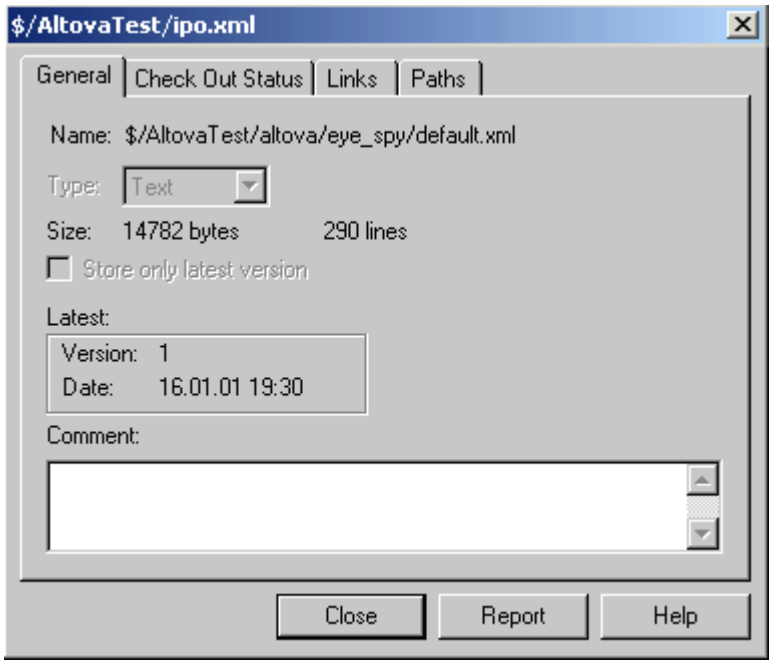


Die Unterschiede zwischen beiden Dateien werden in beiden Fenstern farblich markiert (dieses Beispiel verwendet MS Source-Safe).

Eigenschaften

Mit diesem Befehl können Sie die Eigenschaften der ausgewählten Datei anzeigen. Die Anzeige kann je nach verwendetem Versionskontrollprodukt unterschiedlich sein.

Dieser Befehl kann jeweils nur an einer einzelnen Datei ausgeführt werden.



Status aktualisieren

Mit diesem Befehl **aktualisieren** Sie den Status aller Projektdateien unabhängig von ihrem derzeitigen Status.

Native Oberfläche ausführen

Mit diesem Befehl **starten** Sie die ursprüngliche Oberfläche Ihrer Versionskontroll-Software.

13.3.7 Dateien hinzufügen...

Mit diesem Befehl fügen Sie Dateien in das aktuelle Projekt ein.Verwenden Sie diesen Befehl um Dateien in einen beliebigen Ordner Ihres Projekts einzufügen. Sie können einzelne Dateien oder Dateigruppen (Auswahl mit **Strg + Klick**) im Dialogfeld "Öffnen" hinzufügen. Wenn Sie Dateien in ein Projekt einfügen, werden diese automatisch - nach Erweiterung sortiert - in die entsprechenden Projektordner verteilt (Einstellungen hierzu im Dialogfeld Projekteigenschaften).

13.3.8 URL hinzufügen...

Mit diesem Befehl fügen Sie eine URL in das aktuelle Projekt ein. Bei Verwendung von URLs in einem Projekt wird das Ziel der URL in das Projekt einbezogen. Wenn eine Batch-Operation an einer URL oder einer Datei mit URL ausgeführt wird, sucht XMLSpy 2005 nach dem Dokument aus der URL, und führt die gewünschte Operation aus.

© 2005 Altova GmbH

13.3.9 Aktive Datei hinzufügen

Mit diesem Befehl fügen Sie die aktive Datei in das aktuelle Projekt ein. Wenn Sie eine Datei von der Festplatte oder über eine URL geöffnet haben, können Sie die Datei mit diesem Befehl in das aktuelle Projekt aufnehmen.

13.3.10 Aktive und verwandte Dateien hinzufügen

Mit diesem Befehl fügen Sie das aktive XML-Dokument und alle verwandten Dateien in das Projekt ein. Beim Arbeiten mit einem XML-Dokument, das auf einem Schema oder einer DTD basiert, fügt dieser Befehl nicht nur das XML-Dokument zum Projekt hinzu sondern auch alle verwandten Dateien (z.B. DTDs und sämtliche geparste Entitäten, auf die die DTD referenziert).

Hinweis: Dateien, die durch Processing Instructions referenziert werden (z.B. XSLT-Dateien) werden nicht als verwandte Dateien behandelt.

13.3.11 Projektordner in Projekt einfügen...

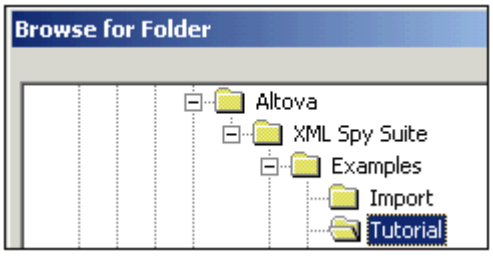
Mit diesem Befehl fügen Sie einen neuen Ordner in das aktuelle Projekt ein. Verwenden Sie diesen Befehl, um einen neuen Ordner oder Unterordner im aktuellen Projekt anzulegen. Sie können diesen Befehl auch über das Kontextmenü (Rechtsklick auf einen Ordner im Projektfenster) aufrufen.

13.3.12 Externes Verzeichnis zum Projekt hinzufügen...

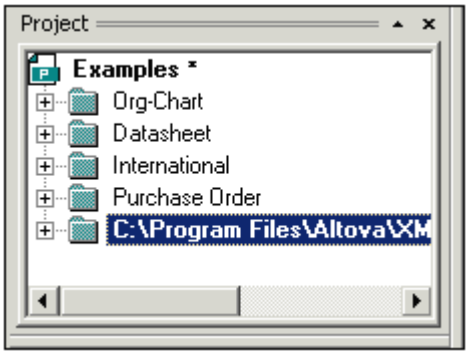
Mit diesem Befehl fügen Sie einen neues **externes Verzeichnis** in das aktuelle Projekt ein. Verwenden Sie diese Funktion, um ein lokales Verzeichnis (oder ein Netzwerkverzeichnis) in das aktuelle Projekt einzufügen. Sie können diesen Befehl auch über das Kontextmenü aufrufen (Rechtsklick auf ein Verzeichnis im Projektfenster). Bitte beachten Sie, dass Dateien in externen Verzeichnissen nicht unter Versionskontrolle gestellt werden können.

So fügen Sie ein externes Verzeichnis in ein Projekt ein:

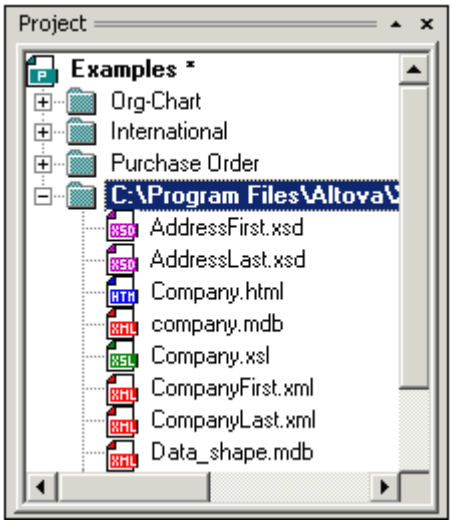
1. Wählen Sie die Menüoption **Projekt | Externes Verzeichnis zu Projekt hinzufügen...**
2. Wählen Sie das Verzeichnis, welches hinzugefügt werden soll (Dialogfeld "Ordner suchen"), und klicken Sie auf OK zum Bestätigen.



Das ausgewählte Verzeichnis wird nun im Projektfenster angezeigt.

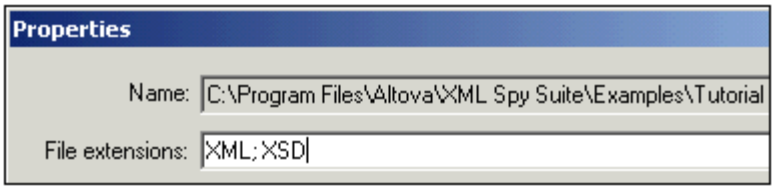


3. Klicken Sie auf das Symbol "+", um den Inhalt des Verzeichnisses anzuzeigen.



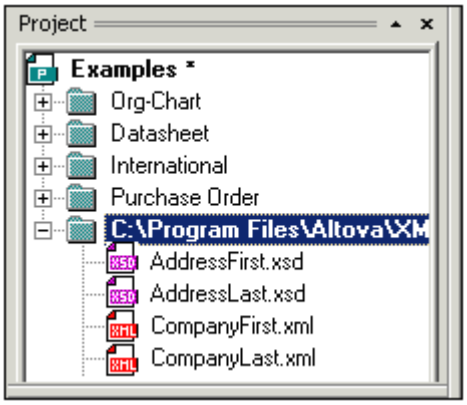
So filtern Sie den Inhalt eines Verzeichnisses:

1. Rechtsklicken Sie auf den lokalen Ordner. Wählen Sie im Popup-Menü die Option **Eigenschaften**.
   Das Dialogfeld Eigenschaften wird geöffnet.



2. Klicken Sie auf das Feld **Erweiterungen**, und geben Sie die Erweiterungen der Dateien an, die angezeigt werden sollen. Bei mehreren Erweiterungen trennen Sie diese durch ein **Semikolon** (in diesem Beispiel XML und XSD Schema).
3. Klicken Sie auf OK um Ihre Auswahl zu bestätigen.

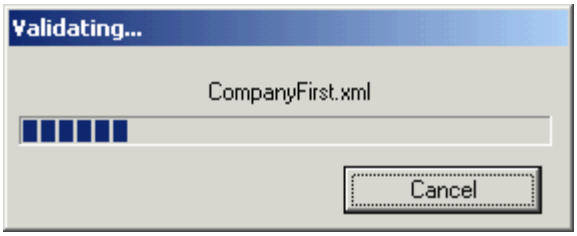
© 2005 Altova GmbH



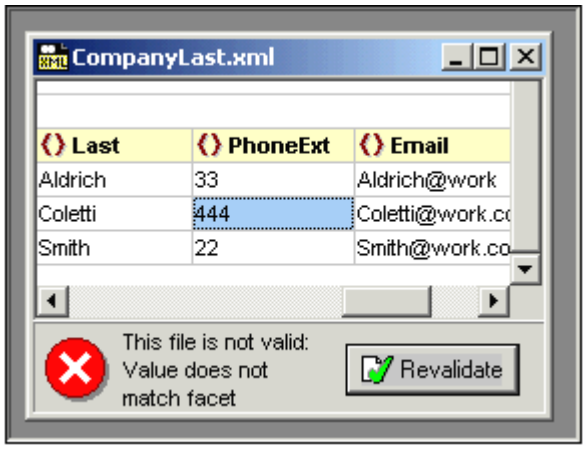
Im Projektfenster werden jetzt ausschließlich XML und XSD Dateien im Verzeichnis "Tutorial" angezeigt.

So prüfen Sie ein Verzeichnis auf Gültigkeit und Wohlgeformtheit:

1. Wählen Sie aus, welche Dateiarten angezeigt werden sollen oder markieren Sie diese im externen Ordner.
2. Klicken Sie auf das Verzeichnis und anschließend auf das Symbol "Wohlgeformtheit prüfen" oder "Validieren" (Kürzel F7, bzw. F8)
   Alle im Ordner angezeigten Dateien werden nun überprüft.



Falls eine Datei nicht wohlgeformt oder ungültig ist, wird diese im Hauptfenster geöffnet. Hier kann der Fehler korrigiert werden. In diesem Beispiel wird die Datei CompanyLast geöffnet, da der Wert PhoneExt nicht mit dem definierten Facet, der im zugrundeliegenden Schema definiert wurde, übereinstimmt. (Das Schema erlaubt nur zweistellige Ziffern für diesen Wert.)



3. Korrigieren Sie den Fehler, und starten Sie die Überprüfung erneut, um das restliche Verzeichnis zu kontrollieren.

Hinweis: Sie können mehrere Dateien gleichzeitig im Verzeichnis markieren, wenn Sie die Strg-

Taste gedrückt halten, und die gewünschten Dateien anklicken. In diesem Fall werden beim Betätigen der Tasten F7 oder F8 nur die ausgewählten Dateien geprüft.

So aktualisieren Sie den Inhalt des Projektverzeichnisses:
In lokalen oder netzwerkgebundenen Verzeichnissen können jederzeit Dateien gelöscht oder hinzugefügt werden. Um die Verzeichnis-Ansicht zu aktualisieren,

- Rechtsklicken Sie auf das externe Verzeichnis, und wählen Sie im Popup-Menü die Option **Externes Verzeichnis aktualisieren**.

Dateien oder Verzeichnisse löschen:

- Um ein Verzeichnis aus dem Projektfenster zu löschen, klicken Sie auf das **Verzeichnis** und drücken Sie die **Entf-**Taste. Das Verzeichnis wird nur aus der Projekt-Ansicht entfernt; auf der Festplatte oder im Netzwerk ist das Verzeichnis weiterhin vorhanden.
- Durch Klick auf eine **einzelne Datei** und Drücken der Entf-Taste wird die Datei **nicht aus dem Projektfenster gelöscht**. Sie müssen sie physisch löschen und dann den Inhalt des externen Ordners aktualisieren.

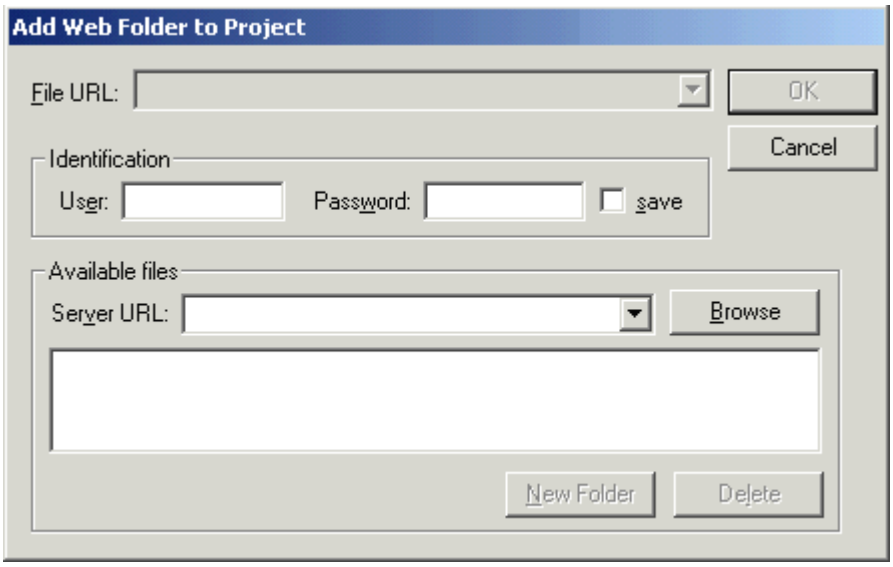
13.3.13 Externes Webverzeichnis zum Projekt hinzufügen...

Mit diesem Befehl fügen Sie ein **externes Webverzeichnis** in das aktuelle Projekt ein.

Verwenden Sie diese Funktion, um ein neues Webverzeichnis in das aktuelle Projekt einzufügen. Sie können diesen Befehl auch über das Kontextmenü (Rechtsklick auf ein Verzeichnis im Projektfenster) aufrufen. Bitte beachten Sie, dass Dateien in externen Verzeichnissen nicht unter Versionskontrolle gestellt werden können.

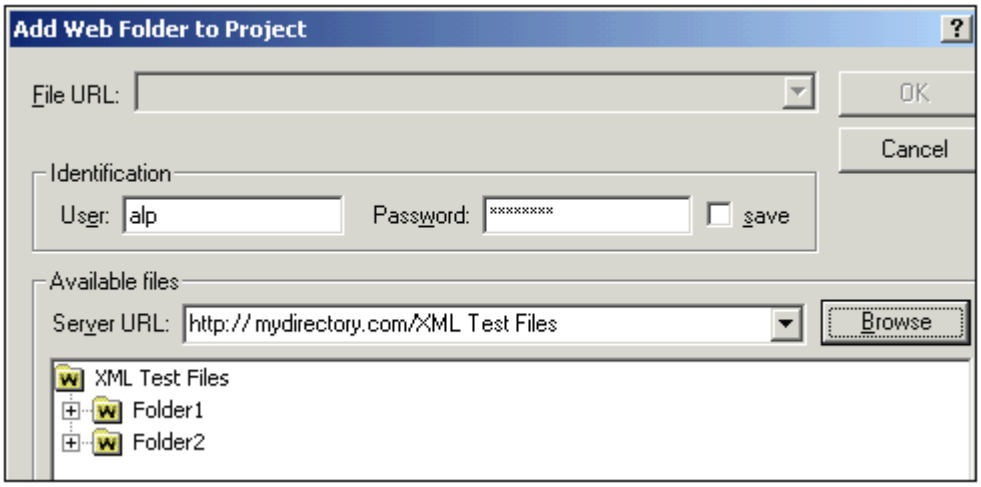
So fügen Sie ein externes Webverzeichnis in ein Projekt ein:

1. Wählen Sie die Menüoption **Projekt | Externes Webverzeichnis zum Projekt hinzufügen**.
   Das Dialogfeld "Webordner in Projekt einfügen" wird angezeigt.

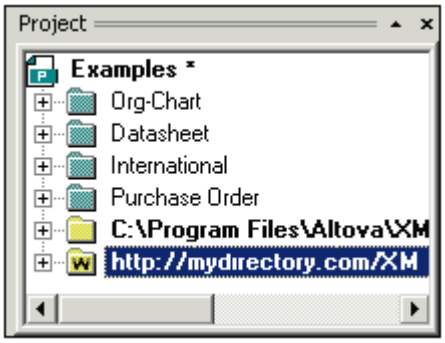


2. Klicken Sie auf das Server URL-Feld, um die URL anzugeben, und geben Sie Benutzernamen und Passwort in die entsprechenden Felder ein.
3. Klicken Sie auf "Durchsuchen" um die Verbindung mit dem Server herzustellen, und die verfügbaren Dateien zu anzuzeigen.

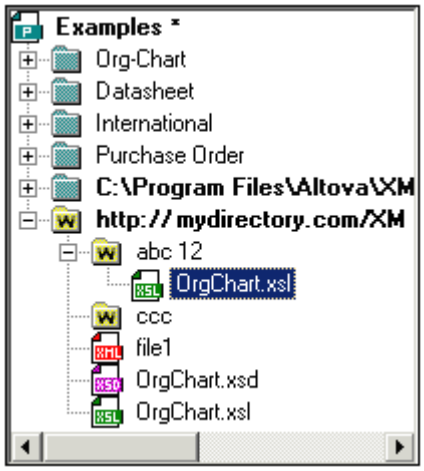
© 2005 Altova GmbH



4. Klicken Sie auf das Verzeichnis, welches Sie in die Projektansicht übernehmen möchten. Die OK Schaltfläche wird erst danach aktiv. Verzeichnisname und http: Serveradresse werden nun im Feld Datei-URL angezeigt.
5. Klicken Sie auf **OK**, um das Verzeichnis in das Projekt zu übernehmen.



6. Klicken Sie auf das Symbol "+" um den Inhalt des Verzeichnisses anzuzeigen.



So filtern Sie den Inhalt eines Verzeichnisses:

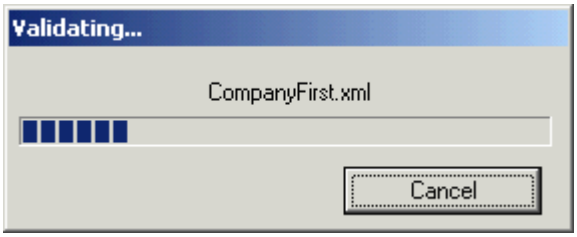
1. **Rechtsklicken** Sie auf das Webverzeichnis, und wählen Sie im Popup-Menü die Option **Eigenschaften**.
   Das Dialogfeld Eigenschaften wird geöffnet.
2. Klicken Sie auf das Feld **Erweiterungen**, und geben Sie die Erweiterungen der Dateien

an, die angezeigt werden sollen. Bei mehreren Erweiterungen trennen Sie diese durch ein **Semikolon** (z.B. XML und Schema XSDs).

3. Klicken Sie auf OK um Ihre Auswahl zu bestätigen.
   Im Projektfenster werden jetzt ausschließlich die XML und XSD Dateien des Webverzeichnisses angezeigt.

So prüfen Sie ein Verzeichnis auf Gültigkeit und Wohlgeformtheit:

1. Klicken Sie auf das Verzeichnis und anschließend auf das Symbol "Wohlgeformtheit prüfen" oder "Validieren" (Kürzel F7, bzw. F8)
   Alle angezeigten Dateien im Verzeichnis werden nun überprüft.



   Falls eine Datei nicht wohlgeformt oder ungültig ist, wird diese im Hauptfenster geöffnet. Hier können Sie den Fehler korrigieren.

2. Nach erfolgter Korrektur, starten Sie die Prüfung erneut, um das restliche Verzeichnis zu kontrollieren.

Bitte beachten Sie:
Sie können mehrere Dateien gleichzeitig im Verzeichnis markieren, wenn Sie die Strg-Taste gedrückt halten, und die gewünschten Dateien anklicken. In diesem Fall werden beim Betätigen der Funktionstasten F7 oder F8 nur die markierten Dateien überprüft.

So aktualisieren Sie den Inhalt des Projektverzeichnisses:
Sie können jederzeit Dateien aus dem Webverzeichnis hinzufügen oder löschen. Um die Verzeichnis-Ansicht zu aktualisieren,

- Rechtsklicken Sie auf das externe Verzeichnis, und wählen Sie im Popup-Menü die Option **Externes Verzeichnis aktualisieren**.

Dateien oder Verzeichnisse löschen:

- Um ein Verzeichnis aus dem Projektfenster zu löschen, klicken Sie auf das **Verzeichnis** und drücken Sie die **Entf-**Taste. Das Verzeichnis wird nur aus der Projekt-Ansicht entfernt; auf der Festplatte oder im Netzwerk ist das Verzeichnis weiterhin vorhanden.
- Durch Klick auf eine **einzelne Datei** und Betätigen der **Entf**-Taste wird die Datei **nicht aus dem Projektfenster gelöscht**. Sie müssen sie physisch löschen, und dann den Inhalt des externen Ordners aktualisieren.

13.3.14 Projekteigenschaften...

Mit diesem Befehl können Sie wichtige Einstellungen für spezifische Ordner Ihres Projekts definieren.

Die Zuweisungen in den Feldern "XSL-Vorlage" bzw. "XML-Vorlage" haben Vorrang vor anderen lokalen Zuweisungen innerhalb einer XML-Datei. So wird z.B. die Datei OrgChart.xsl ("XSL-Vorlage") immer dann verwendet, wenn XML-Dateien im **XML-Dateiverzeichnis**

 © 2005 Altova GmbH

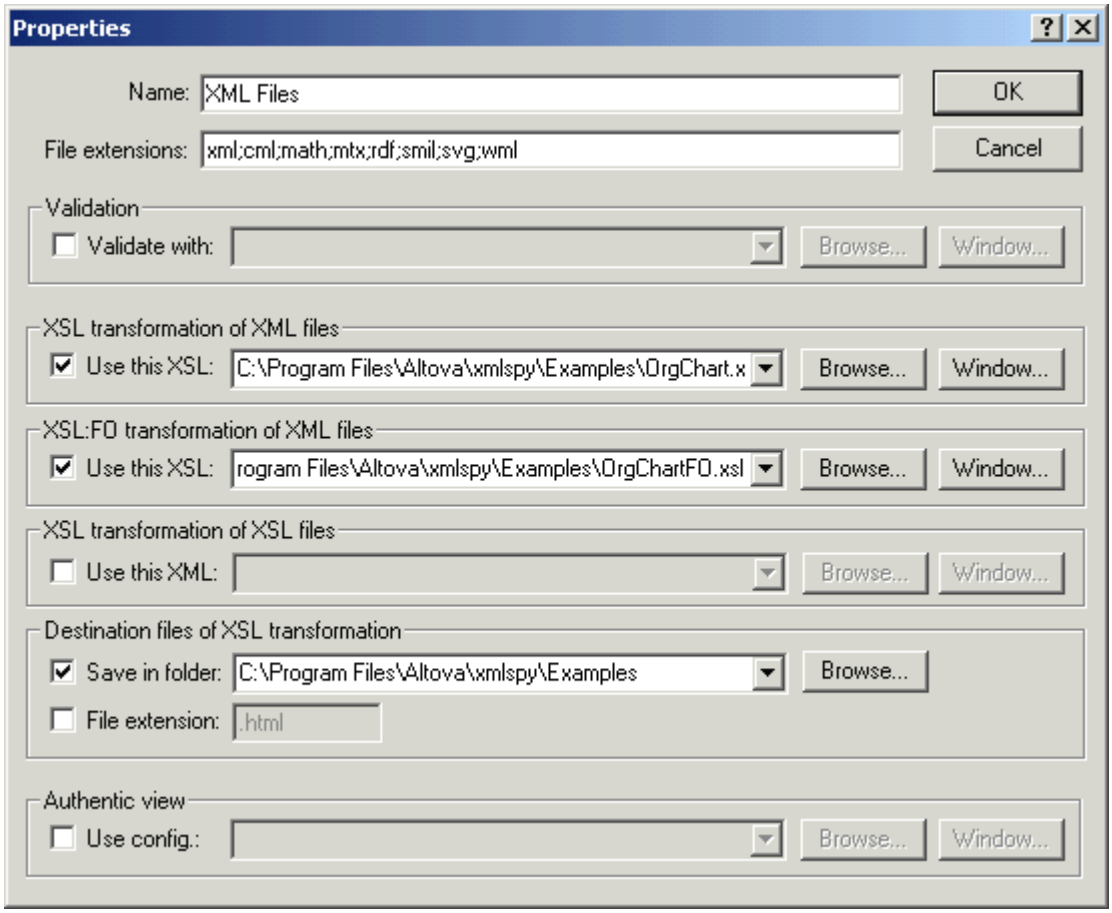
transformiert werden. Die hier vorgenommene Zuweisung hat auch Vorrang vor anderen Zuweisungen, die für übergeordnete Ordner vorgenommen wurden.

So definieren Sie Projekteigenschaften für ein Verzeichnis:

1. Rechtsklicken Sie auf das Verzeichnis, für welches Sie Einstellungen vornehmen möchten, und
2. Wählen Sie den Befehl **Eigenschaften...** aus dem Kontextmenü.

Bitte beachten Sie:

Falls die Projektdatei unter Versionskontrolle steht, werden Sie gefragt, ob die Projektdatei (*.spp) ausgecheckt werden soll. Nach dem Ändern der Einstellungen klicken Sie auf OK um diese zu speichern.



Dateierweiterungen
Die Dateierweiterungen helfen beim Hinzufügen neuer Dateien zum Projekt (im Gegensatz zum Einordnen in einen bestimmten Ordner) bei der automatischen Einordnung der Dateien in Ordner eines Projekts.

Validierung
Definieren Sie das DTD- oder Schemadokument, welches bei der Validierung aller Dateien des aktuellen Verzeichnisses verwendet werden soll (in diesem Beispiel die Hauptseiten).

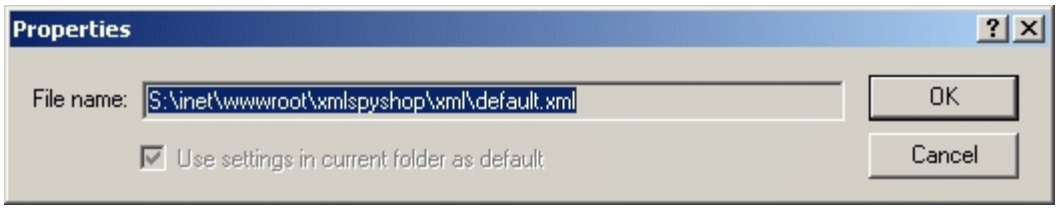
XSL Transformation von XML-Dateien
Definieren Sie das XSL Stylesheet, welches zur XSL Transformation aller Dateien eines Verzeichnisses verwendet werden soll.

Wenn Sie selbst XSL Stylesheets erstellen, können Sie ein XML-Beispieldokument zuweisen, das als Vorschau für das XSL Stylesheet bei der XML-Transformation dient. Sie können dies anstelle des XML Instanzdokuments verwenden.

XSL:FO Transformation von XML-Dateien
Sie können das XSL Stylesheet mit XSL:FO Markup definieren, das für eine XSL:FO Transformation aller Dateien eines Verzeichnisses verwendet werden soll.

Zieldateien bei XSL Transformation
Für Batchtransformationen in XSL können Sie das Zielverzeichnis angeben, in dem die transformierten Dateien abgelegt werden sollen.

Wenn Sie eine Datei oder URL in mehreren Verzeichnissen Ihres Projekts abgelegt haben, können Sie das Standardverzeichnis definieren, mit dessen Einstellungen die Validierung oder Transformation im non-batch Modus erfolgen soll.



Authentic
Die Option "Konfig. verwenden" ermöglicht es Ihnen, das Authentic Stylesheet (SPS Datei) im aktuellen Verzeichnis auszuwählen, welche beim Editieren von XML Dateien in der Authentic-Ansicht zugrunde gelegt werden soll. Nachdem Sie die Schema-, SPS- und XML-Dateien miteinander verknüpft und in ein Projekt eingegeben haben, kann ein Verschieben dieser Dateien in einen anderen Ordner Fehler bei den Verknüpfungen verursachen.

Um solche Fehler zu vermeiden, sollten Sie am besten einen endgültigen Speicherort für Ihre Schema-, SPS- und XML-Dateien festlegen, bevor Sie sie miteinander verknüpfen und einem Projekt zuweisen. Weitere Informationen zur Verwendung der Authentic-Ansicht finden Sie im Handbuch zur Authentic-Ansicht.

13.3.15 Zuletzt verwendete Projekte

Mit dieser Funktion werden die neun zuletzt verwendeten Projekte angezeigt um einen schnelleren Zugriff auf diese zu erhalten.

Beachten Sie bitte, dass XMLSpy 2005 automatisch das zuletzt verwendete Projekt beim Starten von XMLSpy 2005 öffnen kann. (**Extras | Optionen|** Register **Datei**, Projekt | Letztes Projekt beim Programmstart öffnen).

© 2005 Altova GmbH

13.4 Menü "XML"

Das XML-Menü enthält alle nötigen Befehle um mit XML-Dokumenten zu arbeiten. Sie können Elemente einfügen oder anhängen, Elementhierarchien ändern, ein Namespace Präfix für Objekte setzen und XPaths im Zusammenhang mit einzelnen XML-Dokumenten evaluieren.

Zu den am häufigsten verwendeten XML-Funktionen zählen die Überprüfung von Dokumenten auf ihre Wohlgeformtheit und Gültigkeit. Sie finden diese Befehle in diesem Menü.

13.4.1 Einfügen

Der Befehl **Einfügen** ist zwar in allen Menüs aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Er enthält ein Untermenü, über das Sie folgende Möglichkeiten haben:

- die XML-Deklaration und Node-Typen (Attribut, Element, Text, CDATA, Comment, Processing Instruction) in XML-Dokumente einfügen,
- DOCTYPE-Deklarationen und externe DTD-Deklarationen in XML-Dokumente einfügen und
- DTD-Deklarationen (ELEMENT, ATTLIST, ENTITY und NOTATION) in DTD-Dokumente und interne DTD-Deklarationen von XML-Dokumenten einfügen.

Wenn Sie den Cursor über den Befehl **Einfügen** platzieren, erscheint ein Untermenü, das die Datenelemente enthält, die eingefügt werden können. Diese Elemente (*siehe Abbildung oben*) sind nur in der Grid-Ansicht aktiv. Die Datenelemente, die in die aktuelle Auswahl eingefügt werden können, sind aktiv.

Attribut einfügen

Strg+Umschalt+I

Diese Funktion ist nur in der Grid-Ansicht aktiv und fügt ein neues **Attribut** vor dem ausgewählten Datenelement ein. Ein eingefügtes Attribut kann in der Grid-Ansicht einige Zeilen vor dem aktuellen Datenelement angezeigt werden, da Attribute in der Grid-Ansicht unmittelbar nach deren Parent-Element und vor allen Child-Elementen dieses Parent-Elements angezeigt werden.

Element einfügen

Strg+Umschalt+E

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt ein neues **Element** vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird das neue Element vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

Text einfügen

Strg+Umschalt+T

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine neue Textzeile vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird die Textzeile nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

CDATA einfügen

Strg+Umschalt+D

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt einen neuen CDATA-Block vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird der CDATA-Block nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

Comment einfügen

Strg+Umschalt+M

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt einen neuen **Comment** vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird die neue Comment-Zeile nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

XML einfügen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine Zeile für die XML-Deklaration vor dem ausgewählten Datenelement ein. Sie müssen die Child-Attribute der XML-Deklaration und die Werte dieser Attribute einfügen. Eine XML-Deklaration muss ungefähr folgendermaßen aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Hinweis: Da ein XML-Dokument nur eine XML-Deklaration am Anfang der Datei enthalten darf, sollten Sie diesen Befehl nur verwenden, wenn die erste Zeile des Dokuments ausgewählt ist und wenn das Dokument noch keine XML-Deklaration enthält.

Processing Instruction einfügen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine neue Processing Instruction (PI) vor dem ausgewählten Datenelement ein. Falls die Auswahl ein Attribut ist, wird die PI nach dem Attribut und vor dem ersten Child-Element des Parent-Elements des Attributs eingefügt.

DOCTYPE einfügen

Der Befehl **DOCTYPE einfügen** steht in der Grid-Ansicht einer XML-Datei zur Verfügung, wenn der Node der obersten Ebene ausgewählt ist. Sie hängen damit eine DOCTYPE-Deklaration am Beginn des XML-Dokuments an. Sie müssen den Namen des DOCTYPE angeben und dieser

© 2005 Altova GmbH

Name muss mit dem Namen des Dokument-Elements übereinstimmen.

Nach Eingabe des DOCTYPE-Namens können Sie die Deklarationen eingeben, die in der internen DTD-Untermenge verwendet werden sollen.

Hinweis:

- Die DOCTYPE-Deklaration darf nur zwischen der XML-Deklaration und dem XML-Dokument-Element stehen.
- Sie können stattdessen auch den Befehl DTD zuweisen verwenden, mit dem Sie eine DOCTYPE-Anweisung erstellen, die auf ein externes DTD-Dokument referenziert.

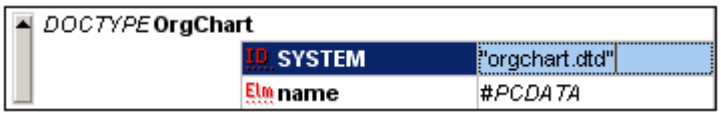
ExternalID einfügen

Eine DOCTYPE-Deklaration in einer XML-Datei kann eine Referenz auf eine externe Ressource, die eine DTD-Deklaration enthält, beinhalten. Diese Ressource wird entweder über einen Public- oder einen System Identifier referenziert.
Beispiel:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

Ein System Identifier ist eine URI, die eine externe Ressource spezifiziert. Der Public Identifier ist ein pfadunabhängiger Identifier, der dazu dient, die Referenz auf eine externe Ressource aufzuheben. So sind z.B. in Ihrem XMLSpy 2005 Installationsverzeichnis URIs für häufig verwendete DTDs und XML-Schemas in einer Katalogdatei mit dem Namen `MainCatalog.xml` gespeichert. Um die Referenz auf eine in `MainCatalog.xml` aufgelistete DTD aufzuheben, können Sie einen Public Identifier verwenden.

Der Befehl **ExternalID einfügen** steht zur Verfügung, wenn in der Grid-Ansicht in einer XML-Datei ein Child-Datenelement der DOCTYPE-Deklaration ausgewählt ist. Mit diesem Befehl fügen Sie in der Grid-Ansicht eine Zeile für einen externen Identifier (`PUBLIC` oder `SYSTEM`) ein. Sie müssen die Art des Identifiers und seinen Wert angeben.



In der Textansicht, die der oben gezeigten Abbildung aus der Grid-Ansicht entspricht, sieht dies aus wie folgt:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Hinweis: Sie können eine Zeile für eine External ID als Child hinzufügen, wenn das Datenelement DOCTYPE ausgewählt ist oder Sie können diese einfügen oder anhängen, wenn eines der Child-Elemente des Datenelements DOCTYPE, z.B. die ELEMENT-Deklaration `name` im obigen Beispiel, ausgewählt ist.

ELEMENT einfügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine ELEMENT-Deklaration vor der ausgewählten Deklaration ein.

ATTLIST einfügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine ATTLIST-Deklaration vor der ausgewählten Deklaration ein.

ENTITY einfügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine ENTITY-Deklaration vor der ausgewählten Deklaration ein.

NOTATION einfügen

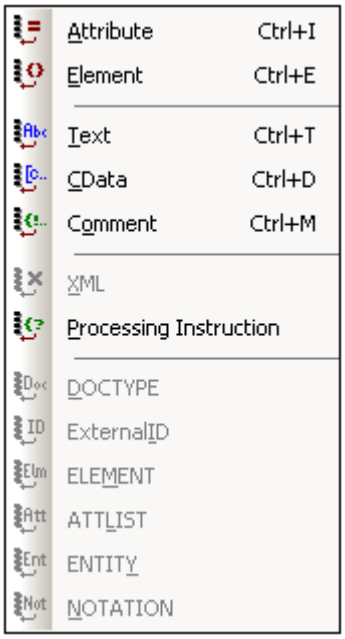
Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl fügt eine NOTATION-Deklaration vor der ausgewählten Deklaration ein.

13.4.2 Anhängen

Der Befehl **Anhängen** ist zwar in allen Menüs aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Er enthält ein Untermenü, über das Sie folgende Möglichkeiten haben:

- die XML-Deklaration und Node-Typen (Attribut, Element, Text, CDATA, Comment, Processing Instruction) in XML-Dokumenten anhängen,
- DOCTYPE-Deklarationen und externe DTD-Deklarationen in XML-Dokumenten anhängen und
- DTD-Deklarationen (ELEMENT, ATTLIST, ENTITY und NOTATION) in DTD-Dokumenten und internen DTD-Deklarationen von XML-Dokumenten anhängen.

© 2005 Altova GmbH



Wenn Sie den Cursor über den Befehl **Anhängen** platzieren, erscheint ein Untermenü, das die Datenelemente enthält, die bei einer bestimmten Auswahl eingefügt werden können. Die Einträge des Untermenüs des Befehls **Anhängen** (*siehe Abbildung oben*) sind nur in der Grid-Ansicht aktiv.

Attribut anhängen

Strg+I

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt ein neues Attribut an.

Element anhängen

Strg+E

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt ein neues **Element** nach dem letzten Geschwisterelement des ausgewählten Elements an. Falls es sich bei der aktuellen Auswahl um einen Attribut-Node handelt, wird der Element-Node nach dem letzten Child des Parent-Elements des Attributs angehängt.

Text anhängen

Strg+T

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen neuen Textblock nach dem letzten Geschwisterelement des ausgewählten Elements an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der Textblock nach letzten Child des Parent-Elements des Attributs angehängt.

CDATA anhängen

Strg+D

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen CDATA Node nach dem letzten Geschwisterelement eines ausgewählten Node (mit Ausnahme eines Attribut-Node) an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der CDATA-Abschnitt nach letzten Child des Parent-Elements des ausgewählten Attributs angehängt.

Comment anhängen

Strg+M

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen Comment Node nach dem letzten Geschwisterelement eines ausgewählten Node (mit Ausnahme eines Attribut-Node) an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der Comment Node nach letzten Child des Parent-Elements des ausgewählten Attributs angehängt.

XML anhängen

Mit dieser Funktion fügen Sie eine neue XML-Deklaration `<?xml version="1.0" encoding="UTF-8"?>` als das erste Datenelement in einem Dokument ein. Bitte beachten Sie, dass ein XML-Dokument nur eine XML-Deklaration enthalten darf, welche am Beginn der Datei stehen muss.

Processing Instruction anhängen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und hängt einen Processing Instruction Node nach dem letzten Geschwisterelement eines ausgewählten Node (mit Ausnahme eines Attribut-Node) an. Falls es sich bei der Auswahl um einen Attribut-Node handelt, wird der Processing Instruction Node nach letzten Child des Parent-Elements des ausgewählten Attributs angehängt.

DOCTYPE anhängen

Der Befehl **DOCTYPE anhängen** steht in der Grid-Ansicht einer XML-Datei zur Verfügung, wenn der Node der obersten Ebene ausgewählt ist. Sie hängen damit eine DOCTYPE-Deklaration am Beginn des XML-Dokuments an. Sie müssen den Namen des DOCTYPE angeben und dieser Name muss mit dem Namen des Dokument-Elements übereinstimmen.

© 2005 Altova GmbH

Nach Eingabe des DOCTYPE-Namens können Sie die Deklarationen eingeben, die in der internen DTD-Untermenge verwendet werden sollen.

Hinweis:

- Die DOCTYPE-Deklaration darf nur zwischen der XML-Deklaration und dem XML-Dokument-Element stehen.
- Sie können stattdessen auch den Befehl DTD zuweisen verwenden, mit dem Sie eine DOCTYPE-Anweisung erstellen, die auf ein externes DTD-Dokument referenziert.

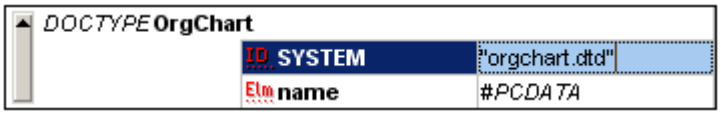
ExternalID anhängen

Eine DOCTYPE-Deklaration in einer XML-Datei kann eine Referenz auf eine externe Ressource, die eine DTD-Deklaration enthält, beinhalten. Diese Ressource wird entweder über einen Public- oder eine System Identifier referenziert.
Beispiel:

```
<!DOCTYPE doc_element_name PUBLIC "publicID" "systemID">
<!DOCTYPE doc_element_name SYSTEM "systemID">
```

Ein System Identifier ist eine URI, die eine externe Ressource spezifiziert. Der Public Identifier ist ein pfadunabhängiger Identifier, der dazu dient, die Referenz auf eine externe Ressource aufzuheben. So sind z.B. in Ihrem XMLSpy 2005 Installationsverzeichnis URIs für häufig verwendete DTDs und XML-Schemas in einer Katalogdatei mit dem Namen `MainCatalog.xml` gespeichert. Um die Referenz auf eine in `MainCatalog.xml` aufgelistete DTD aufzuheben, können Sie einen Public Identifier verwenden.

Der Befehl **ExternalID anhängen** steht zur Verfügung, wenn in der Grid-Ansicht in einer XML-Datei ein Child-Datenelement der DOCTYPE-Deklaration ausgewählt ist. Mit diesem Befehl fügen Sie in der Grid-Ansicht eine Zeile für einen externen Identifier (`PUBLIC` oder `SYSTEM`) ein. Sie müssen die Art des Identifiers und seinen Wert angeben.



In der Textansicht, die der oben gezeigten Abbildung aus der Grid-Ansicht entspricht, sieht dies aus wie folgt:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Hinweis: Sie können eine Zeile für eine External-ID als Child hinzufügen, wenn das Datenelement DOCTYPE ausgewählt ist oder Sie können diese einfügen oder anhängen, wenn eines der Child-Elemente des Datenelements DOCTYPE, z.B. die ELEMENT-Deklaration `name` im obigen Beispiel, ausgewählt ist.

ELEMENT anhängen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine ELEMENT-Deklaration an die Liste der Deklarationen an.

ATTLIST anhängen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine ATTLIST-Deklaration an die Liste der Deklarationen an.

ENTITY anhängen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine ENTITY-Deklaration an die Liste der Deklarationen an.

NOTATION anhängen

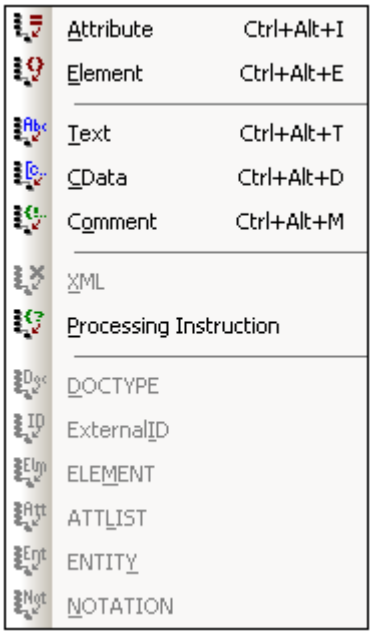
Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn in der DOCTYPE-Deklaration eines XML-Dokuments ein Datenelement ausgewählt ist. Der Befehl hängt eine NOTATION-Deklaration an die Liste der Deklarationen an.

13.4.3 Child hinzufügen

Der Befehl **Child hinzufügen** ist zwar in allen Menüs aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Der Befehl enthält ein Untermenü, über das Sie die folgenden Child-Datenelemente zum aktuell ausgewählten Element hinzufügen können:

- die XML-Deklaration und Node-Typen (Attribut, Element, Text, CDATA, Comment, Processing Instruction) in XML-Dokumenten,
- DOCTYPE-Deklarationen und externe DTD-Deklarationen in XML-Dokumenten und
- DTD-Deklarationen (ELEMENT, ATTLIST, ENTITY und NOTATION) in DTD-Dokumenten und internen DTD-Deklarationen von XML-Dokumenten.

© 2005 Altova GmbH



Wenn Sie den Cursor über den Befehl **Child hinzufügen** platzieren, erscheint ein Untermenü, das die Datenelemente enthält, die eingefügt werden können. Die Einträge des Untermenüs des Befehls **Child hinzufügen** (*siehe Abbildung oben*) sind nur in der Grid-Ansicht aktiv.

Child-Attribut hinzufügen

Strg+ALT+I

Dieser Befehl steht nur in der Grid-Ansicht und nur bei Auswahl eines Element-Node zur Verfügung. Er fügt ein **neues Attribut** als Child des ausgewählten Element-Node ein.

Child-Element hinzufügen

Strg+ALT+E

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt ein neues Element als Child des ausgewählten Node ein.

Child-Text hinzufügen

Strg+ALT+T

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt ein neuen Textinhalt als Child des ausgewählten Datenelements ein.

Child CDATA hinzufügen

Strg+ALT+D

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt einen neuen CDATA-Abschnitt als Child des ausgewählten Datenelements ein.

Child Comment hinzufügen

Strg+ALT+M

Dieser Befehl steht nur in der Grid-Ansicht zur Verfügung. Er fügt einen neuen Comment Node als Child des ausgewählten Datenelements ein.

Child XML hinzufügen

Dieser Befehl ist nur in der Grid-Ansicht und nur, wenn die Datei **leer** ist, verfügbar. Er fügt eine neue XML-Deklaration `<?xml version="1.0" encoding="UTF-8"?>` als das erstes Datenelement in ein Dokument ein. Bitte beachten Sie, dass ein XML-Dokument nur eine XML-Deklaration enthalten darf, welche am Beginn der Datei stehen muss.

Child Processsing Instruction hinzufügen

Diese Funktion steht nur in der Grid-Ansicht zur Verfügung und fügt eine neue Processing Instruction (PI) als Child des ausgewählten Datenelements ein.

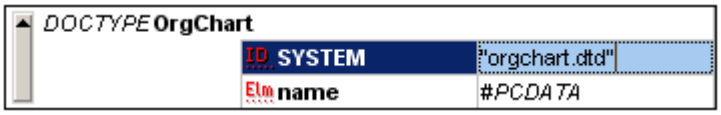
Child DOCTYPE hinzufügen

Dieser Befehl steht nur in der Grid-Ansicht und nur bei einem **leeren** Dokument zur Verfügung. Er fügt eine DOCTYPE-Deklaration in ein XML-Dokument ein. Mit Hilfe der DOCTYPE-Deklaration können Sie eine interne DTD-Untermenge deklarieren.

Child ExternalID hinzufügen

Der Befehl **ExternalID hinzufügen** steht nur dann zur Verfügung, wenn in der Grid-Ansicht in einer XML-Datei die DOCTYPE-Deklaration ausgewählt ist. Mit diesem Befehl fügen Sie in der Grid-Ansicht eine Zeile für einen externen Identifier (`PUBLIC` oder `SYSTEM`) ein. Sie müssen die Art des Identifiers und seinen Wert angeben.

© 2005 Altova GmbH



In der Textansicht, die der oben gezeigten Abbildung aus der Grid-Ansicht entspricht, sieht dies aus wie folgt:

```
<!DOCTYPE OrgChart SYSTEM "orgchart.dtd" [
 <!ELEMENT name (#PCDATA)>
]>
```

Hinweis: Sie können eine Zeile für eine External-ID als Child hinzufügen, wenn das Datenelement DOCTYPE ausgewählt ist oder Sie können diese einfügen oder anhängen, wenn eines der Child-Elemente des Datenelements DOCTYPE, z.B. die ELEMENT-Deklaration `name` im obigen Beispiel, ausgewählt ist.

Child ELEMENT hinzufügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine ELEMENT-Deklaration an die Liste der Deklarationen an.

Child ATTLIST hinzufügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine ATTLIST-Deklaration an die Liste der Deklarationen an.

Child ENTITY hinzufügen

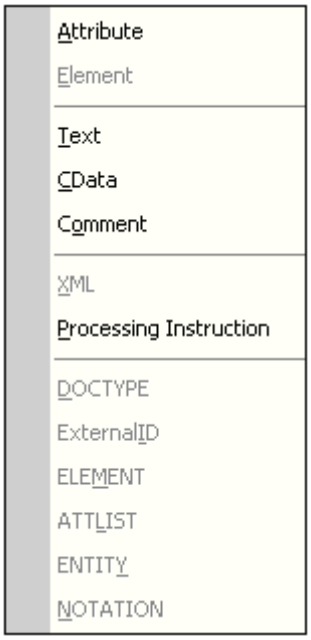
Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine ENTITY-Deklaration an die Liste der Deklarationen an.

Child NOTATION hinzufügen

Diese Funktion steht nur in der Grid-Ansicht und nur für DTD-Dokumente zur Verfügung oder, wenn die DOCTYPE-Deklaration eines XML-Dokuments ausgewählt ist. Der Befehl hängt eine NOTATION-Deklaration an die Liste der Deklarationen an.

13.4.4 Konvertieren in

Mit diesem Befehl können Sie ein ausgewähltes Datenelement in der Grid-Ansicht in einen anderen Datenelement-Typ konvertieren. Diese Operation steht nur in der Grid-Ansicht für einzelne Datenelemente zur Verfügung, die keinen Child Node enthalten. Wenn Sie den Cursor über den Befehl **Konvertieren in** platzieren, wird ein Untermenü mit den Datenelementen angezeigt, in die das ausgewählte Datenelement konvertiert werden kann.



Gehen durch die Konvertierung möglicherweise Daten verloren (z.B. geht beim Konvertieren eines Attributs in einen Comment der Attributname verloren), werden Sie mittels einer Meldung darüber informiert.

In Attribut konvertieren

Das ausgewählte Datenelement wird in ein **Attribut** konvertiert.

In Element konvertieren

Das ausgewählte Datenelement wird in ein **Element** konvertiert.

In Text konvertieren

Das ausgewählte Datenelement wird in **Text** konvertiert.

In CDATA konvertieren

Das ausgewählte Datenelement wird in einen **CDATA Block** konvertiert.

© 2005 Altova GmbH

In Comment konvertieren

Das ausgewählte Datenelement wird in einen **Comment** konvertiert.

In XML konvertieren

Mit dieser Funktion konvertieren Sie ein ausgewähltes Datenelement in eine XML Deklaration

`<?xml version="1.0" encoding="UTF-8"?>`.

Hinweis: Jedes XML-Dokument darf nur eine XML-Deklaration enthalten, die am Beginn der Datei stehen muss.

In Processing Instruction konvertieren

Das ausgewählte Datenelement wird in eine neue **Processing Instruction** (PI) konvertiert.

In DOCTYPE konvertieren

Mit diesem Befehl konvertieren Sie das ausgewählte Datenelement in eine DOCTYPE-Deklaration (in einer XML-Datei) für die interne Untermenge einer Document Type Declaration (DTD).

Hinweis: Die DOCTYPE-Deklaration darf nur am Beginn eines XML-Instanzdokuments zwischen der XML-Deklaration und dem Dokument-Element des XML-Dokuments stehen.

In ExternalID konvertieren

Das ausgewählte Datenelement wird in eine externe DTD-Referenz in einer DOCTYPE-Deklaration (`PUBLIC` oder `SYSTEM` Identifier) konvertiert.

In ELEMENT konvertieren

Das ausgewählte Datenelement wird in eine Element-Deklaration in einer DOCTYPE-Deklaration oder einer externen DTD konvertiert.

In ATTLIST konvertieren

Das ausgewählte Datenelement wird in eine Attributlisten-Deklaration in einer DOCTYPE Deklaration oder einer externen DTD konvertiert.

In ENTITY konvertieren

Das ausgewählte Datenelement wird in eine Entity-Deklaration in einer DOCTYPE Deklaration oder einer externen DTD konvertiert.

In NOTATION konvertieren

Das ausgewählte Datenelement wird in eine Notationsdeklaration in einer DOCTYPE Deklaration oder einer externen DTD konvertiert.

13.4.5 Tabelle

Der Befehl **Tabelle** ist zwar in allen Ansichten aktiv, kann aber nur in der Grid-Ansicht verwendet werden. Er enthält ein Untermenü mit allen für die Datenbank/Tabellen-Ansicht der Grid-Ansicht relevanten Befehle.

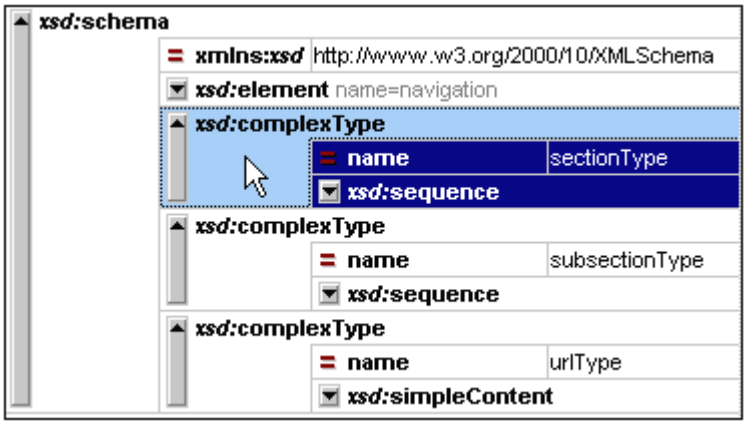
Als Tabelle anzeigen

F12

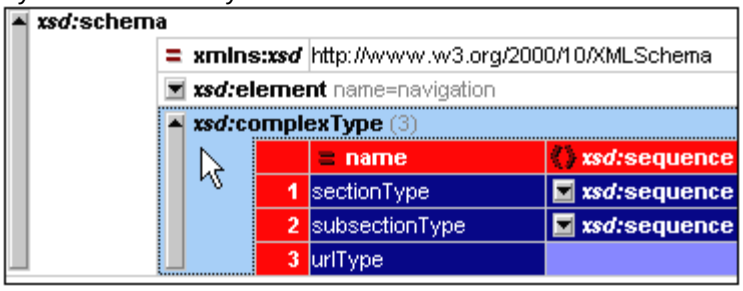
Mit diesem Befehl können Sie zwischen der Grid-Standardansicht und der Datenbank/Tabellen-Ansicht (oder Tabellenansicht) eines Dokument-Elements wechseln. In der Tabellenansicht werden sich wiederholende Elemente als Tabelle angezeigt, wobei die Zeilen die Instanzen enthalten und die Spalten die Child Nodes (einschl. Comments, CDATA-Abschnitte und PIs).

Um zur Tabellenansicht zu wechseln, gehen Sie vor wie folgt:

1. Markieren Sie ein sich wiederholendes Element, das in Tabellenform angezeigt werden soll.



2. Wählen Sie **XML | Als Tabelle anzeigen**, drücken Sie **F12** oder klicken Sie auf das Symbol in der Symbolleiste.



Das Element wird als Tabelle dargestellt und das Symbol ist aktiv.

Um von der Tabellenansicht eines Dokument-Elements zur normalen Grid-Ansicht dieses Elements zu wechseln, wählen Sie die Tabelle oder eine ihrer Zeilen oder Spalten aus und klicken Sie in der Symbolleiste auf das Symbol. Daraufhin wird das Tabellenelement in der

© 2005 Altova GmbH

Grid-Ansicht angezeigt.

Zeile einfügen

Umschalt+F12

Der Befehl **Zeile einfügen** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Zeile oder Zelle ausgewählt ist. Der Befehl fügt eine neue Zeile vor der markierten Zeile ein. Die neue Zeile entspricht einer Instanz des Tabellenelements. Für das neue Element werden obligatorische Child-Elemente erstellt.

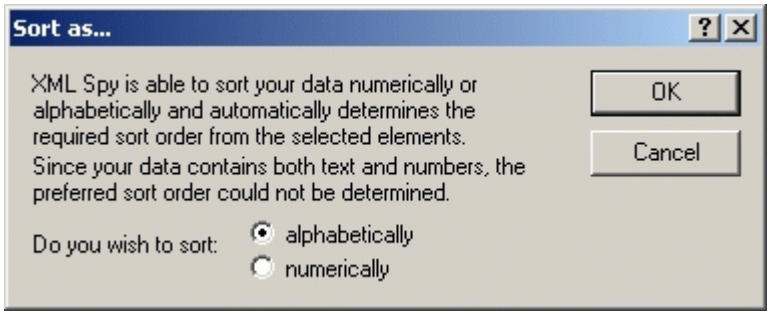
Zeile anhängen

Strg+F12

Der Befehl **Zeile anhängen** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Zeile oder Zelle ausgewählt ist. Der Befehl hängt eine neue Zeile nach der letzten Zeile der Tabelle an. Die neue Zeile entspricht einer Instanz des Tabellenelements. Für das neue Element werden obligatorische Child-Elemente erstellt.

Aufsteigend sortieren

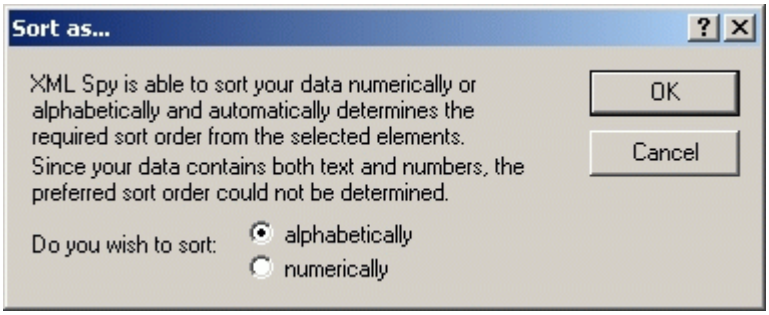
Der Befehl **Aufsteigend sortieren** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Spalte oder Zelle ausgewählt ist. Der Befehl sortiert die Spalte entweder alphabetisch oder numerisch in aufsteigender Reihenfolge. XMLSpy 2005 versucht automatisch zu ermitteln, um welche Art von Daten es sich handelt und sortiert diese alphabetisch oder numerisch. Wenn vom System nicht erkannt wird, um welche Art es sich handelt, werden Sie aufgefordert die Sortiermethode anzugeben (*siehe Abbildung unten*).



Absteigend sortieren

Der Befehl **Absteigend sortieren** ist in der Datenbank/Tabellen-Ansicht aktiv, wenn eine Spalte oder Zelle ausgewählt ist. Der Befehl sortiert die Spalte entweder alphabetisch oder numerisch in absteigender Reihenfolge. XMLSpy 2005 versucht automatisch zu ermitteln, um welche Art von Daten es sich handelt und sortiert diese alphabetisch oder numerisch. Wenn vom System nicht erkannt wird, um welche Art es sich handelt, werden Sie aufgefordert die Sortiermethode

anzugeben (*siehe Abbildung unten*).



13.4.6 Höher stufen

Strg+L

Der Befehl **Höher stufen** steht nur in der Grid-Ansicht zur Verfügung. Sie verschieben damit den markierten Node um eine Ebene nach links, d.h. ein Child-Element wird hierdurch zu einem Geschwister-Element seines Parent. Dieser Befehl wird auch "**Promote"** genannt.

13.4.7 Tiefer stufen

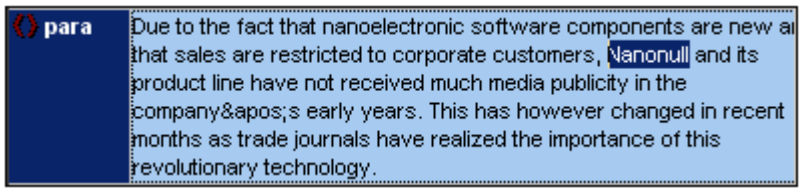
Strg+R

Der Befehl **Tiefer stufen** steht nur in der Grid-Ansicht zur Verfügung. Sie verschieben damit den markierten Node um eine Ebene nach rechts, sodass er hierdurch zu einem Child-Element seines ehemaligen Geschwister-Element wird. Dieser Befehl wird auch "**Demote"** genannt.

13.4.8 In Element einbinden

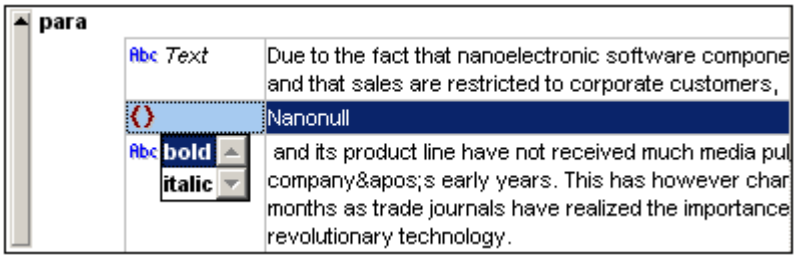
Der Befehl **In Element einbinden** steht nur in der Grid-Ansicht zur Verfügung. Er bindet einen markierten Textbereich in ein neues Element ein. Das neue Element wird innerhalb des Textbereichs rund um den markierten Text angelegt. Wenn Sie ein Dokument auf Basis eines Schemas oder einer DTD bearbeiten, steht Ihnen automatisch eine Liste mit gültigen Auswahlmöglichkeiten für den Namen des Elements, in das der Text eingeschlossen werden soll, zur Verfügung.

In der unten gezeigten Abbildung ist z.B. der Text `Nanonull` im Element `para` markiert.



Wenn Sie den Befehl **XML | In Element einbinden** auswählen, wird der Text "Nanonull" in ein neues Element eingeschlossen, das innerhalb des Texts erstellt wird. Es wird eine Liste angezeigt, aus der Sie für den Elementnamen `bold` oder `italic` auswählen können. Diese Elemente werden im Schema als Child-Elemente von `para` definiert.

© 2005 Altova GmbH

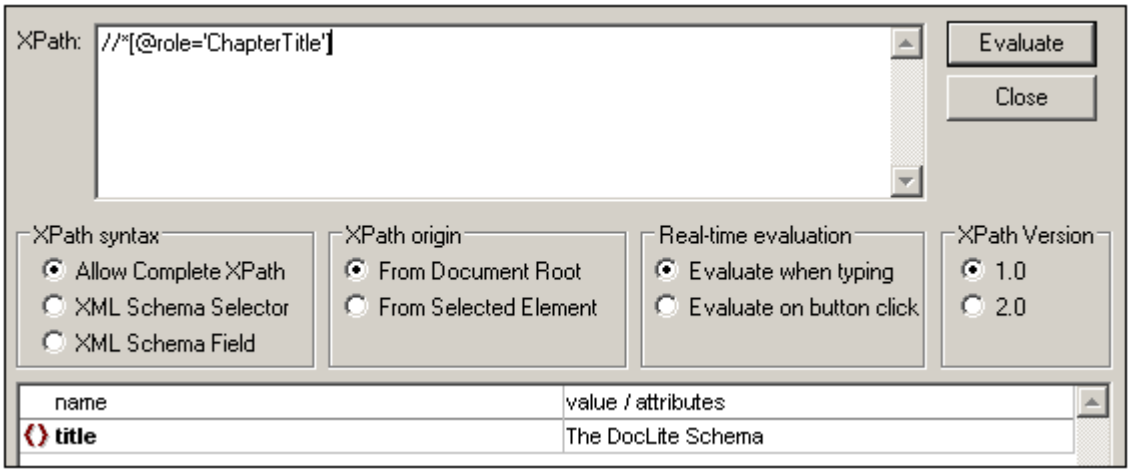


Der gewählte Eintrag wird als Name des neuen Elements verwendet. Alternativ dazu können Sie auch einen anderen Namen für das Element eingeben.

13.4.9 XPath evaluieren

Bei Auswahl des Befehls **XPath evaluieren...** erscheint das Dialogfeld XPath evaluieren (*siehe unten*). In diesem Dialogfeld können Sie XPath 1.0- oder XPath 2.0-Ausdrücke für das aktive XML-Dokument überprüfen. Der Befehl **XPath evaluieren...** ist in der Text- und der Grid-Ansicht aktiv.

Das Dialogfeld "XPath evaluieren" besteht aus drei Bereichen: einem Eingabebereich, einem Bereich mit Optionen und einem Ergebnisbereich. Der XPath-Ausdruck wird in den Eingabebereich eingegeben und das Ergebnis der Überprüfung wird im Ergebnisbereich angezeigt. Als Ergebnis erscheint eine Liste von Elementen in der Ergebnissequenz zusammen mit den entsprechenden Werten. Wenn Sie ein Element in der Liste auswählen, erscheint dieses im Hauptfenster markiert.



XPath-Ausdruck im Eingabebereich
Beachten Sie bitte die folgenden allgemeinen Punkte zu XPath-Ausdrücken. XPath 1.0- und XPath 2.0-spezifische Punkte werden weiter unten gesondert erörtert.

- Wenn Sie im Eingabebereich einen XPath-Ausdruck eingeben, wird er schwarz angezeigt, wenn die Syntax korrekt ist, und rot, wenn sie nicht korrekt ist. Beachten Sie, dass eine korrekte Syntax nicht bedeutet, dass der Ausdruck keinen Fehler enthält. Fehlermeldungen werden im Ergebnisbereich angezeigt.
- Aufgrund von falsch eingestellten XPath Evaluator-Optionen kann es zu Fehlern

kommen. Achten Sie daher darauf, dass diese Optionen korrekt eingestellt sind, v.a. die XPath-Version und der XPath-Ausgangspunkt.
- Es werden nur Namespaces evaluiert, die sich bei dem Element, das den XPath-Ausgangspunkt bildet (also dem Kontext-Node), im Geltungsbereich befinden.
- Der XPath Evaluator verwendet für die beiden XPath-Versionen unterschiedliche Prozessoren, daher gibt es keine Probleme mit der Rückwärtskompatibilität.

XPath 1.0 Ausdrücke

- XPath 1.0-Funktionen müssen ohne Namespace-Präfix eingegeben werden.
- Die vier Node-Tests nach Typ werden unterstützt: `node()`, `text()`, `comment()`und `processing-instruction()`.

XPath 2.0 Ausdrücke

- Strings (z.B. 'Hello') und numerische Literale (z.B. 256) werden unterstützt. Um andere, auf XML-Schema-Typen basierende Literale zu erstellen, verwenden Sie einen Konstruktor mit einem Namespace-Präfix (z.B. `xs:date('2004-09-02')`). Das Namespace-Präfix, das Sie für XML-Schema-Typen verwenden, muss an den XML-Schema-Namespace gebunden sein: `http://www.w3.org/2001/XMLSchema`. Dieser Namespace muss in Ihrer XML-Datei deklariert sein.
- XPath 2.0-Funktionen, die vom XPath Evaluator verwendet werden, gehören zum Namespace `http://www.w3.org/2004/10/xpath-functions`. Konventionsgemäß ist das Präfix `fn:` an diesen Namespace gebunden. Da dieser Namespace aber der Namespace der Standardfunktionen ist, die vom XPath Evaluator verwendet werden, sollten Sie bei Funktionen **kein** Präfix angeben. Falls Sie dies trotzdem tun, stellen Sie sicher, dass das Präfix an den Namespace der XPath 2.0-Funktionen gebunden ist, den Sie im XML-Dokument deklarieren müssen. Beispiel für die Verwendung von Funktionen: `current-date()` (wobei der Namespace der Funktionen nicht in der XML-Datei deklariert ist); `fn:current-date()` (wobei der Namespace der Funktionen in der XML-Datei deklariert ist und an das Präfix `fn:` gebunden ist). Sie können das Namespace-Präfix selbst dann weglassen, wenn der Namespace der Funktionen in der XML-Datei deklariert ist, da sich die Funktion in diesem Fall im Default Namespace (also dem Namespace der Funktionen) befindet..
- Bei Verwendung der beiden XDT-Datentypen (`yearMonthDuration` und `dayTime Duration`), muss der XDT Namespace (`http://www.w3.org/2004/10/xpath-datatypes`) in der XML-Datei deklariert sein und das Namespace-Präfix muss bei diesen Typen im XPath-Ausdruck verwendet werden. Beispiel: `years-from-yearMonthDuration(xdt:yearMonthDuration('P22Y18M')`.

Hinweis: Zusammenfassend ist zur Namespace-Frage zu sagen: Wenn Sie Konstruktoren aus dem XML-Schema-Namespace oder Typen aus dem XDT-Namespace verwenden, müssen Sie die Namespaces im XML-Dokument deklarieren und die korrekten Namespace-Präfixe im XPath-Ausdruck verwenden. Für XPath-Funktionen ist kein Präfix erforderlich.

Datentypen in XPath 2.0
Wenn Sie einen XPath 2.0-Ausdruck für ein XML-Dokument evaluieren, das ein XML-Schema referenziert und gemäß diesem Schema gültig ist, müssen Sie Datentypen, die nicht implizit durch eine Operation in den erforderlichen Datentyp konvertiert werden, explizit konstruieren oder casten. Im XPath 2.0-Datenmodell, das vom integrierten XPath-Prozessor verwendet wird, wird allen **atomisierten** Node-Werten aus dem XML-Dokument der `xdt:untypedAtomic` Datentyp zugewiesen. Der `xdt:untypedAtomic` Typ funktioniert gut bei impliziten Typkonvertierungen. So ergibt z.B. der Ausdruck `xdt:untypedAtomic("1") + 1` den Wert 2, weil der `xdt:untypedAtomic` Wert implizit durch das Plus-Zeichen auf `xs:double` höher

 © 2005 Altova GmbH

gestuft wird. Arithmetische Operatoren stufen Operanden implizit zu `xs:double` höher. Vergleichsoperatoren stufen Operanden vor dem Vergleich zu `xs:string` höher.

In einigen Fällen ist es jedoch notwendig, den Datentyp explizit in den benötigten zu konvertieren. Wenn Sie z.B. zwei Elemente, `startDate` und `endDate` verwenden, die im XML-Schema als Typ `xs:date` definiert sind, wird bei Verwendung des XPath 2.0-Ausdrucks `endDate - startDate` ein Fehler angezeigt. Wenn Sie jedoch `xs:date(endDate) - xs:date(startDate)` oder `(endDate cast as xs:date) - (startDate cast as xs:date)` verwenden, wird der Ausdruck korrekt auf eine einzige Sequenz vom Typ `xdt:dayTimeDuration` evaluiert.

Hinweis: Die vom XPath Evaluator verwendeten Prozessoren werden auch vom Altova XSLT-Prozessor verwendet, daher müssen XPath 2.0-Ausdrücke in XSLT Stylesheets, die nicht implizit in den benötigten Datentyp konvertiert werden, explizit als der benötigte Datentyp konstruiert oder gecastet werden.

XPath Evaluator-Optionen

Zur Verfügung stehen die folgenden Evaluierungsoptionen:

- XPath-Syntax: Bei XPath 1.0-Überprüfungen können Sie auswählen, ob die vollständige XPath 1.0 Grammatik, die XML Schema Selector-Grammatik oder die XML Schema Field Grammatik verwendet werden soll.
- XPath-Ausgangspunkt: Soll der Kontext Node der Dokument-Node (oder die Dokument-Root) oder ein ausgewählter Node sein.
- Echtzeitevaluierung: Sollen die Ergebnisse der Evaluierung während der Eingabe angezeigt werden oder wenn Sie auf die Schaltfläche **Validieren** klicken.
- XPath-Version: Wählen Sie aus, ob zur Überprüfung des Ausdrucks XPath 1.0 oder XPath 2.0 verwendet werden soll.

Hinweis: Die Optionen werden auch für zukünftige Evaluierungen beibehalten, bis sie geändert werden.

Fehlerbehebung

Wird in Ihrem XPath-Ausdruck ein Fehler gefunden, gehen Sie vor wie folgt:

1. Überprüfen Sie, ob die Optionen korrekt eingestellt wurden.
2. Überprüfen Sie, ob die Namen von Funktionen, Konstruktoren, Nodes usw. richtig geschrieben sind.
3. Überprüfen Sie, ob Präfixe erforderlich sind und korrekt für Funktionen, Konstruktoren und Argumente im XPath-Ausdruck eingestellt wurden.
4. Ist im XML-Dokument ein Namespace deklariert, überprüfen Sie die Richtigkeit der Namespaces. Die Namespaces sollten lauten wie folgt:

   XML Schema: `http://www.w3.org/2001/XMLSchema`
   XPath Datentypen (XDT): `http://www.w3.org/2004/10/xpath-datatypes`
   XPath 2.0 Funktionen: `http://www.w3.org/2004/10/xpath-functions`

Unterstützung von XPath 2.0-Funktionen

Nachstehend sehen Sie eine Liste von vordefinierten XPath 2.0 Funktionen, die nur beschränkt unterstützt werden.

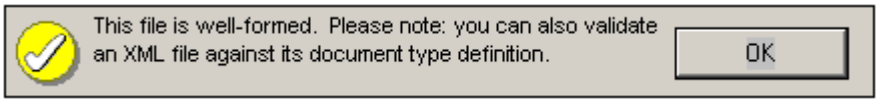
Funktion	Eingeschränkte Unterstützung
`fn:id`	Nicht unterstützt.
`fn:idref`	Nicht unterstützt.
`fn:lower-case`	Gilt nur für den lateinischen Zeichensatz.
`fn:normalize-unicode`	Nicht unterstützt.
`fn:upper-case`	Gilt nur für den lateinischen Zeichensatz.

13.4.10 Wohlgeformtheit prüfen

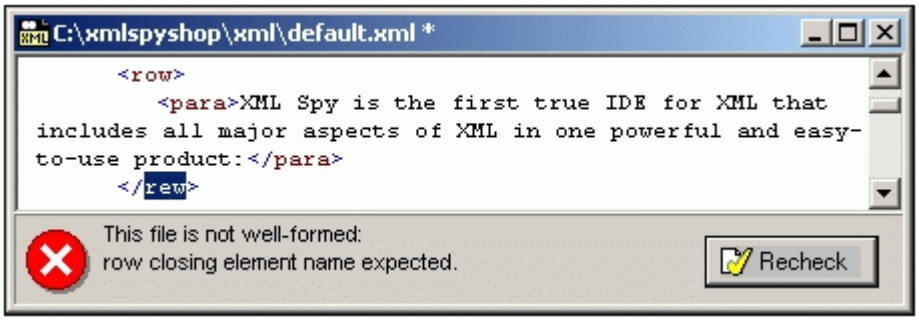
F7

Mit diesem Befehl überprüfen Sie das Dokument auf Wohlgeformtheit nach der Definition der XML 1.0 Spezifikation. Jedes XML-Dokument muss wohlgeformt sein; deswegen überprüft XMLSpy 2005 ein Dokument automatisch auf Wohlgeformtheit, sobald es geöffnet oder gespeichert wird, oder wenn eine andere als die Text-Ansicht ausgewählt wird. Sie können das Dokument auch während der Bearbeitung jederzeit auf Wohlgeformtheit prüfen.

Bei erfolgreicher Wohlgeformtheitsprüfung wird am unteren Rand des Hauptfensters eine Meldung angezeigt:



Wenn bei der Wohlgeformtheitsprüfung ein Fehler entdeckt wurde, wird das Fehler verursachende Objekt markiert und eine entsprechende Fehlermeldung am unteren Rand des Hauptfensters angezeigt.



Eigentlich ist es nicht erlaubt, ein schlecht geformtes XML-Dokument zu speichern; XMLSpy 2005 enthält die Funktion "Save anyway" (trotzdem speichern), die angezeigt wird, wenn die automatische Wohlgeformtheitsprüfung beim Speichern des Dokument einen Fehler entdeckt. Dies ist nützlich, wenn sie Ihre Arbeit kurzzeitig unterbrechen müssen, und den Fehler nicht sofort beheben können.

Hinweis: Sie können den Befehl **Wohlgeformtheit prüfen** auch an jeder Datei, jedem Ordner oder jeder Gruppe von Dateien im aktiven Projektfenster anwenden. Klicken Sie auf den entsprechenden Ordner, und anschließend auf das Symbol "Wohlgeformtheit prüfen".

© 2005 Altova GmbH

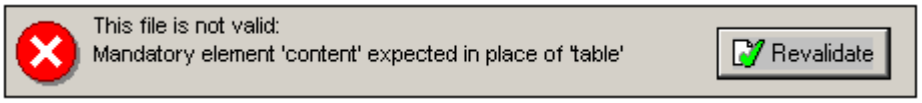
13.4.11 Validieren

F8

Mit diesem Befehl können Sie ein XML-Dokument gegen DTDs, XML-Schemas und andere Schemas validieren. Die Validierung wird automatisch durchgeführt, wenn Sie von der Textansicht in eine andere Ansicht wechseln. Sie können einstellen, dass ein Dokument beim Öffnen oder Speichern automatisch validiert wird (**Extras | Optionen | Datei**). Der Befehl **Validieren** führt vor Überprüfung der Gültigkeit gleichzeitig eine Wohlgeformtheitsprüfung durch, sodass diese vorher nicht separat überprüft werden muss.

Bei erfolgreicher Validierung wird am unteren Rand des Hauptfensters folgende Meldung angezeigt:

Wenn bei der Validierung ein Fehler entdeckt wurde, wird das Fehler verursachende Objekt markiert, und eine entsprechende Fehlermeldung wird angezeigt:



Der Befehl wird normalerweise am aktiven Dokument durchgeführt. Sie können die Validierung über Dateien, Ordner oder Dateigruppen laufen lassen, die im aktiven Projektfenster angezeigt werden. Klicken Sie im Projektfenster auf die gewünschte Datei/den gewünschten Ordner und wählen Sie die Option **XML | Validieren** oder **F8.** Ungültige Dateien werden als aktive Datei im Hauptfenster geöffnet und die entsprechende Fehlermeldung wird angezeigt.

Validieren von XML-Dokumenten
Um eine XML-Datei zu validieren, machen Sie das XML-Dokument im Haupfenster zum aktiven und klicken Sie auf **XML | Validieren** oder drücken Sie **F8**. Das XML-Dokument wird daraufhin gegen das in der XML-Datei referenzierte Schema validiert. Existiert keine Referenz, wird am unteren Rand des Hauptfensters eine Fehlermeldung angezeigt. Solange das XML-Dokument geöffnet ist, bleibt das Schema im Speicher (siehe Cache-Speicher leeren im DTD/Schema-Menü).

Validieren von Schema-Dokumenten (DTDs und XML-Schema)
XMLSpy 2005 unterstützt die gebräuchlichen Schemadialekte einschließlich DTD und XML Schema. Um ein Schema-Dokument zu validieren, machen Sie es im Hauptfenster aktiv und klicken Sie auf **XML | Validieren** oder drücken Sie **F8**.

XMLSpy 2005 unterstützt einen Teil der OASIS XML-Kataloge. Über diese Kataloge kann XMLSpy 2005 gebräuchliche Schemas (sowie Stylesheets und andere Dateien) aus lokalen Benutzerordnern abrufen, wodurch die Verarbeitungsgeschwindigkeit erhöht wird, Benutzer auch offline arbeiten können (d.h. ohne mit einem Netzwerk verbunden zu sein) und Dokumente leichter übertragbar werden (da die URIs nur in den Katalogdateien geändert werden müssen). Der Mechanismus funktioniert folgendermaßen:

- Eine Katalogdatei ordnet einen `PUBLIC` System Identifier einer URI zu, die auf eine lokale Datei verweist. In XMLSpy 2005 durchsuchen die folgenden beiden Dateien den Katalog: `MainCatalog.xml` und `CustomCatalog.xml`. `MainCatalog.xml` ordnet

vordefinierte PUBLIC System Identifier mehrerer gängiger Schemas URIs zu, die auf das lokal gespeicherte Schema verweisen. CustomCatalog.xml erlaubt Ihnen, Ihre eigenen Katalog-Extensions zu erstellen. Sowohl MainCatalog.xml als auch CustomCatalog.xml wird gemeinsam mit Ihrer XMLSpy 2005 Applikation installiert. Die Schemas, Stylesheets und anderen Dateien, die von MainCatalog.xml referenziert werden, befinden sich in bestimmten Unterordnern Ihres XMLSpy 2005 Installationsordners.

- Der PUBLIC Identifier in der DOCTYPE-Anweisung Ihrer XML-Datei wird für das Durchsuchen des Katalogs verwendet. Bei häufig verwendeten Schemas ist der PUBLIC Identifier normalerweise bereits vordefiniert, sodass nur mehr die URI in den Katalog-Lookup-Dateien geändert werden muss, wenn XML-Dokumente auf mehreren Rechnern verwendet werden. In XMLSpy 2005 wird zuerst in MainCatalog.xml gesucht und anschließend in CustomCatalog.xml.

Wenn Sie Ihre eigene CustomCatalog.xml Datei schreiben, verwenden Sie nur die folgende Teilmenge des OASIS-Katalogs, damit XMLSpy 2005 den Katalog korrekt verarbeiten kann.

```
<catalog...>
<public publicId="nnn" uri="mmm"/>
<uri name="nnn" uri="mmm"/>
<system systemId="nnn" uri="mmm"/>
<rewriteURI uriIdStartString="nnn" rewritePrefix="mmm"/>
<rewriteSystem systemIdStartString="nnn" rewritePrefix="mmm"/>
```

Hinweis:

- Die DTDs für XML-Schemas werden zwar von MainCatalog.xml aus referenziert, doch können Sie Ihre XML-Dateien gegen keines dieser Schemas validieren. Zweck dieser beiden DTDs ist es, zu Bearbeitungszwecken für die Eingabehilfen Informationen bereitzustellen, falls Sie Dateien gemäß diesen älteren Empfehlungen erstellen wollen.
- Nähere Informationen zu Katalogen finden Sie in der XML Catalogs specification.

13.4.12 Eingabehilfen aktualisieren

Mit diesem Befehl aktualisieren Sie die Eingabehilfen-Fenster. Dies erfolgt durch erneutes Laden der zugrunde liegenden DTD / des zugrunde liegenden Schemas.
Sollten Sie das Schema oder die DTD, auf dem/auf der ein offenes XML-Dokument basiert, verändert haben, empfiehlt es sich, das Schema oder die DTD erneut zu laden, damit die intelligenten Editierinformationen im XML-Dokument aktualisiert werden.

13.4.13 Namespace Präfix...

Dieser Befehl steht in der Grid-Ansicht zur Verfügung und öffnet ein Dialogfeld, in dem Sie das Namespace Präfix des ausgewählten Elements oder Attributs - und im Fall von Elementen - seiner Nachfahren definieren können.

© 2005 Altova GmbH

Sie können das Namespace Präfix für Elemente, Attribute oder für beide setzen. Das Namespace Präfix wird auf das ausgewählte Element oder Attribut angewendet und - wenn ein Element ausgewählt ist - auch auf die untergeordneten Nodes des ausgewählten Elements.

13.5 Menü "DTD/Schema"

Das **DTD/Schema**-Menü enthält Befehle für das effiziente Arbeiten mit DTDs und XML-Schemas.

Dieser Abschnitt enthält eine umfassende Beschreibung aller Befehle in diesem Menü.

13.5.1 DTD zuweisen...

Dieser Befehl ist aktiv, wenn eine XML-Datei aktiv ist. Sie weisen damit einem XML-Dokument eine DTD zu, wodurch Validierung und intelligentes Editieren ermöglicht wird. Wenn Sie diesen Befehl wählen, wird das Dialogfeld zum Zuweisen einer Datei geöffnet, in dem Sie die gewünschte DTD zuweisen können. Beachten Sie, dass Sie den Pfad der zugewiesenen Datei relativ machen können. Aktivieren Sie dazu das Kontrollkästchen "Pfad relativ zu.. machen". Nach erfolgter Zuweisung enthält Ihr Dokument eine DOCTYPE-Deklaration, die die zugewiesene DTD referenziert. Die DOCTYPE-Deklaration sieht ungefähr folgendermaßen aus:

```
<!DOCTYPE main SYSTEM"http://link.xmlspy.com/spyweb.dtd">
```

Beachten Sie, dass eine DTD einer neuen XML-Datei zum Zeitpunkt der Erstellung zugewiesen werden kann.

13.5.2 Schema zuweisen...

Dieser Befehl ist aktiv, wenn eine XML-Datei aktiv ist. Sie weisen damit einem XML-Dokument ein XML-Schema zu, wodurch Validierung und intelligentes Editieren ermöglicht wird. Wenn Sie diesen Befehl wählen, wird das Dialogfeld zum Zuweisen einer Datei geöffnet, in dem Sie das gewünschte XML-Schema zuweisen können. Beachten Sie, dass Sie den Pfad der zugewiesenen Datei relativ machen können. Aktivieren Sie dazu das Kontrollkästchen "Pfad relativ zu... machen". Nach erfolgter Zuweisung enthält Ihr Dokument eine XML-Schema-Zuweisung mit dem benötigten Namespace. Die Schema-Zuweisung sieht ungefähr folgendermaßen aus:

```
      xmlns="http://www.xmlspy.com/schemas/icon/orgchart"
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xsi:schemaLocation="http://www.xmlspy.com/schemas/icon/orgchart
      http://schema.xmlspy.com/schemas/icon/orgchart.xsd"
```

Die mit XMLSpy 2005 erzeugten Deklarationen hängen vom Schematyp und von der potentiellen Verwendung eines targetNamespace im Schemadokument ab.

13.5.3 Weitere DTD einfügen...

Mit diesem Befehl können Sie eine weitere Dokument Type Definition (DTD) oder extern geparste Entity in das interne Subset einer DTD, oder in jede andere DTD inkludieren. Dies geschieht durch Definition einer entsprechenden extern geparsten Entity-Deklaration und Verwendung dieser Entity in folgender Zeile:

```
<!ENTITY % navigation.dtd SYSTEM"S:\xml\navigation.dtd">
%navigation.dtd;
```


© 2005 Altova GmbH

Mit diesem Befehl öffnen Sie das Dialogfeld zum Zuweisen einer Datei, in dem Sie angeben können, welche DTD in Ihre DTD inkludiert werden soll.

Hinweis: Dieser Befehl ist nur in der Grid-Ansicht aktiv.

13.5.4 Gehe zu DTD

Mit diesem Befehl öffnen Sie die DTD, auf der das aktive XML-Dokument basiert. Wurde keine DTD zugewiesen, wird eine Fehlermeldung angezeigt.

13.5.5 Gehe zu Schema

Mit diesem Befehl öffnen Sie das XML-Schema, auf dem das aktive XML-Dokument basiert. Wurde kein XML-Schema zugewiesen, wird eine Fehlermeldung angezeigt.

13.5.6 Gehe zu Definition

Mit diesem Befehl wird die exakte Definition eines Elements oder Attributs in der zugehörigen DTD bzw. dem zugehörigen Schema-Dokument angezeigt.

So zeigen Sie die Definition des Datenelements in der Enhanced Grid Ansicht an:

1. Klicken Sie auf das Datenelement.
2. Wählen Sie im Menü **DTD/Schema | Gehe zu Definition**, oder klicken Sie auf das oben abgebildete Symbol.

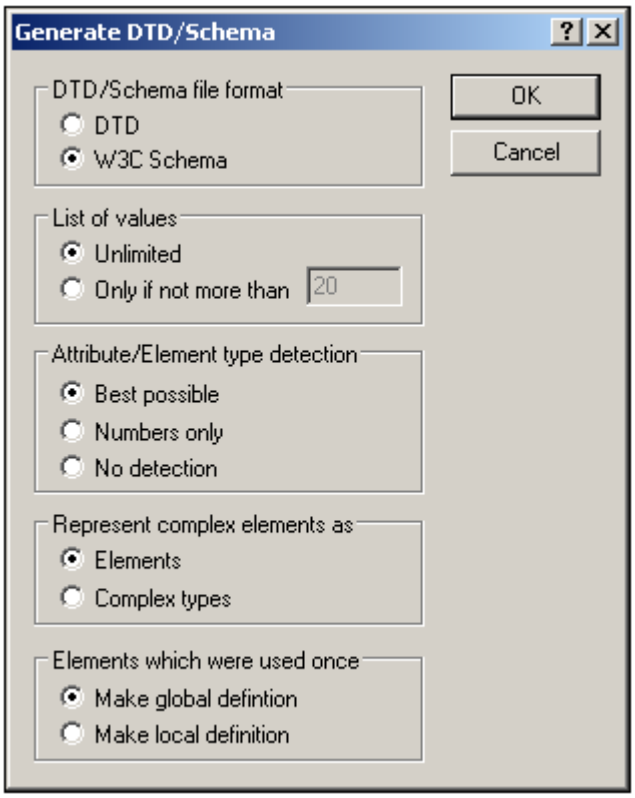
So zeigen Sie die Definition des Datenelements in der Schema/WSDL Design-Ansicht an:

1. Mit Strg + Doppelklick auf das Datenelement, dessen Definition Sie ansehen möchten, oder
2. Klicken Sie auf das Objekt und wählen Sie die Menüoption **DTD/Schema | Gehe zu Definition,** oder klicken Sie auf das Symbol.

   In beiden Fällen wird die zugehörige DTD- oder Schemadatei geöffnet und die Definition des Datenelements wird markiert.

13.5.7 DTD/Schema generieren...

Mit diesem Befehl wird eine neue DTD oder ein W3C XML-Schema von einem XML-Dokument (oder von einer Gruppe von Dokumenten im Projekt Fenster) erzeugt.
Verwenden Sie diese Funktion, wenn Sie ein neues Schema oder eine neue DTD anhand eines XML-Dokuments erzeugen möchten.



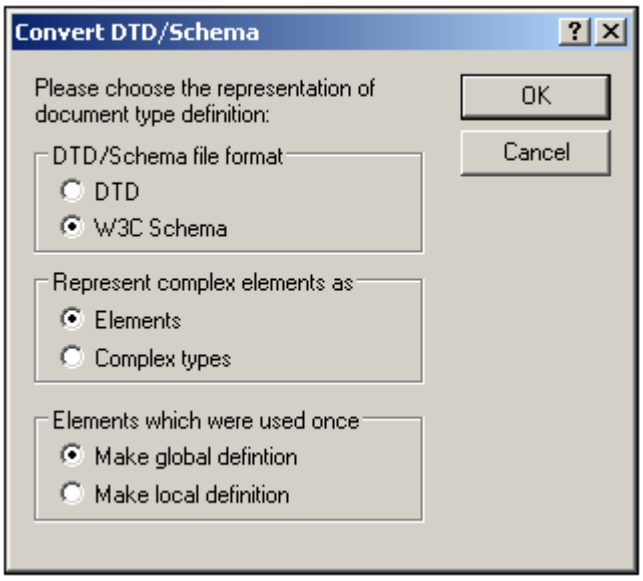
Wenn Sie ein XML-Schema generieren, erkennt XMLSpy 2005 automatisch die Datentypen, die in Ihrem XML-Dokument verwendet werden, und erstellt die nötigen Typen im Schema. XMLSpy 2005 kann optional auch Enumerationsszenarien erkennen (Listenwerte), in denen Elemente oder Attribute nur bestimmte Werte aus einer vordefinierten Werteliste enthalten dürfen. Weiters können Sie entscheiden, wie komplexe Elemente dargestellt werden sollen, oder wie Elemente, die nur einmal vorkommen, behandelt werden sollen.

Dieser Befehl wird normalerweise im aktiven Hauptfenster durchgeführt, allerdings können Sie ihn auch auf jede Datei, Ordner oder Gruppe von Dateien im aktiven Projekt Fenster anwenden.

13.5.8 DTD/Schema konvertieren...

Mit diesem Befehl konvertieren Sie eine existierende DTD in ein XML-Schema und umgekehrt. Bei Auswahl der Option "W3C Schema" müssen Sie außerdem angeben, wie Complex-Elemente (als Elemente oder Complex Types) und Elemente, die nur einmal verwendet werden (global oder lokal) dargestellt werden sollen.

© 2005 Altova GmbH



Kovertieren einer DTD in ein XML-Schema
Bei der Konvertierung einer DTD in ein XML-Schema macht XMLSpy 2005 aufgrund der beschränkten Informationen, die zur Verfügung stehen, einige Annahmen. Beachten Sie vor allem, dass die Werte bestimmter DTD-Komponenten wörtlich interpretiert werden, anstatt dass ihre Semantik geparst wird, da das Programm nicht weiß, welche der vielen Möglichkeiten die richtige ist. Bearbeiten Sie die generierte Konvertierung in diesem Fall.

Das Resultat sollte jedenfalls immer genau überprüft und gegebenenfalls verbessert werden. Im Folgenden werden einige Bereiche angegeben, in denen Sie eventuell Korrekturen vornehmen sollten.

Definition von Datentypen für Attribute
DTDs unterstützen nur 10 Attribut-Datentypen, während in XML-Schemas z.B. 44 Datentypen plus davon abgeleitete Datentypen zulässig sind. Durch Verwendung von weniger Datentypen lässt sich ein generiertes XML-Schema unter Umständen verbessern. Beachten Sie, dass Datentyp-Informationen bei der Konvertierung eines XML-Schemas in eine DTD verloren gehen.

Namespaces
DTD erkennen keine Namespaces. Wenn daher in einer DTD Namespaces definiert werden, müssen sie in Form von Element- und Attributnamen hartkodiert werden. Bei der Konvertierung von einem Schema in ein anderes kann dies einige Probleme verursachen.

Entities
In XML-Schemas gibt es kein Äquivalent für die allgemeinen Entity-Deklarationen von DTDs. Wenn XMLSpy 2005 eine DTD in ein XML-Schema konvertiert, ignoriert es die Entity-Deklarationen.

Nicht geparste Datendeklarationen
DTDs und XML-Schemas verwenden bei der Behandlung nicht geparster Daten unterschiedliche Mechanismen. Dies wird im Folgenden näher erklärt.

DTDs verwenden den folgenden Mechanismus:

1. Eine Notation wird aus einem Namen und einem Identifier bestehend deklariert, z.B.
   `<!NOTATION gif SYSTEM "image/gif">`
2. Sie deklarieren die Entity, z.B.
   `<!ENTITY cover_img SYSTEM "graphics/cover_img.gif" NDATA gif>`

3. Normalerweise gibt man zum betreffenden Attribut einen Attributtyp von ENTITY an, z.B.
```
<!ELEMENT img EMPTY>
<!ATTLIST img format ENTITY #REQUIRED>
```

In einem XML-Schema wird hierfür der folgende Mechanismus verwendet:

1. Deklarieren Sie eine Notation. Dies funktioniert auf dieselbe Art wie bei der DTD.
```
<xs:notation name="gif" public="image/gif"/>
```
Beachten Sie, dass das Attribut `public` obligatorisch ist und den Identifier enthält. Ein optionales Attribut `system` enthält den System Identifier und ist normalerweise eine ausführbare Datei, die Ressourcen des Notationstyps verarbeiten kann.
2. Man verknüpft die Notationsdeklaration über den NOTATION-Datentyp mit einem bestimmten Attributwert.
   - Sie können den `NOTATION`-Datentyp jedoch nicht direkt verwenden, sondern müssen einen weiteren Datentyp aus dem NOTATION-Datentyp ableiten.
```
<xs:simpleType name="formatType">
  <xs:restriction base="xs:NOTATION">
    <xs:enumeration value="gif"/>
    <xs:enumeration value="jpeg"/>
  </xs:restriction>
</xs:simpleType>
```
   - Man verknüpft das Attribut mit dem vom `NOTATION`-Datentyp abgeleiteten Datentyp, z.B.
```
<xs:complexType name="imgType">
  <xs:attribute name="height"/>
  <xs:attribute name="width"/>
  <xs:attribute name="location"/>
  <xs:attribute name="format" type="formatType"
  use="required"/>
</xs:complexType>
<xs:element name="img" type="imgType"/>
```

Beim Konvertieren einer DTD in ein XML-Schema mit Hilfe des Befehls "DTD/Schema konvertieren..." geht XMLSpy 2005 vor wie folgt:

- Die folgende Deklaration:
```
<!ATTLIST image format ENTITY #REQUIRED
...>
```
  wird in
```
<xs:attribute name="format" type="xs:ENTITY" use="required"/>
```
  konvertiert.

- und
```
<!NOTATION gif SYSTEM "image/gif">
```
  wird in
```
<xs:notation name="gif" system="image/gif"/>
```
  konvertiert.

Nehmen Sie daher die folgenden Änderungen vor:

1. In Notationen wie `<xs:notation name="gif" system="image/gif"/>` ersetzen Sie `system` durch `public` und fügen einen optionalen System Identifier hinzu, falls erforderlich.
2. Leiten Sie aus dem `NOTATION` Datentyp einen Datentyp ab, wie oben für `formatType` beschrieben.

© 2005 Altova GmbH

3. Verknüpfen Sie den abgeleiteten Datentyp mit dem entsprechenden Attribut.

Hinweis: Gemäß der XML-Schemaspezifikation, ist es nicht notwendig - bzw. je nach Sichtweise, nicht möglich - eine externe Entity zu deklarieren.

Konvertieren eines XML-Schemas in eine DTD
Bei der Konvertierung eines XML-Schemas in eine DTD werden die im XML-Schema verwendeten **Namespace-Präfixe** - nicht die Namespace URIs oder die Namespace-Deklarationen - in die Namen der entsprechenden Elemente und Attribute in der DTD übernommen. Beachten Sie die folgenden Punkte:

1. Da XML-Parser Namespaces beim Validieren eines Dokuments gegen eine DTD ignorieren, werden die Namespace-Deklarationen selbst nicht konvertiert.
2. Die Attribute `elementFormDefault` und `attributeFormDefault` des Elements `xs:schema` bestimmen, von welchen Elementen und Attributen die Präfixe im Konvertierungsvorgang übernommen werden. Ist der Wert auf `unqualified` gesetzt, werden nur die Präfixe von global deklarierten Elementen bzw. Attributen im Konvertierungsprozess übernommen. Ist der Wert auf `qualified` gesetzt, werden die Präfixe aller Element- und Attributnamen in den Konvertierungsprozess eingeschlossen.
3. Präfixe werden in den entsprechenden Stringwert plus Doppelpunkt konvertiert. Elemente und Attribute in Default Namespaces werden in Elemente und Attribute konvertiert, deren Namen mit: `default_NS_X` beginnt, wobei X eine Ganzzahl ist (beginnend mit 1 bis zu einem Maximalwert, der der Anzahl der im XML-Schema verwendeten Default Namespaces entspricht).
4. In der DTD sind Elementnamen aus Parameter-Entities zusammengesetzt. Auf diese Art können Sie das Präfix in der DTD jederzeit ändern, falls das Präfix im XML-Dokument einmal geändert werden muss. Parameter-Entity-Definitionen können entweder in der DTD selbst oder durch Außerkraftsetzung der Parameter-Entity-Definitionen im internen DTD Subset des XML-Dokuments geändert werden.

Hinweis: Namespaces haben keinen semantischen Wert in einer DTD. Namespace-Präfixe, die aus dem XML-Schema übernommen werden, werden einfach nur Teil des Namens des in der DTD definierten Elements oder Attributs.

13.5.9 In MapForce auf andere(s) DTD/Schema oder DB mappen...

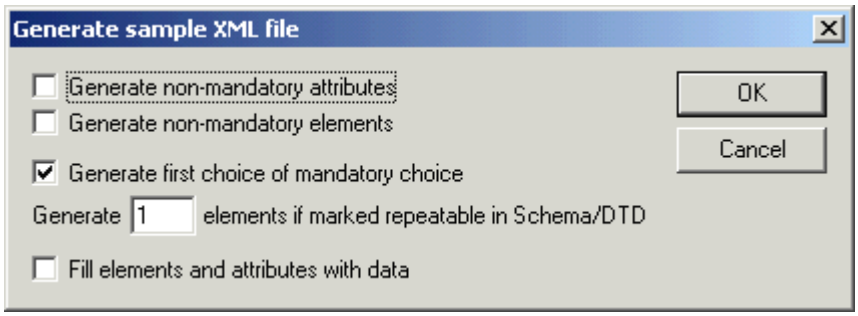
Startet Altova MapForce 2005, wenn die Applikation installiert ist. In MapForce 2005 können Sie ein Schema auf eine andere DTD, ein anderes XML-Schema oder eine Datenbank mappen.

13.5.10 HTML/PDF-Ausgabe in StyleVision erstellen

Startet Altova StyleVision 2005, wenn die Applikation installiert ist. In StyleVision 2005 können Sie Stylesheets für die Ausgabe von Dokumenten im HTML-, PDF- und RTF-Format erstellen.

13.5.11 XML Beispieldatei erzeugen...

Dieser Befehl erstellt eine XML-Datei, die auf dem aktiven Schema (DTD oder XML-Schema) im Hauptfenster basiert.



Nicht obligatorische Attribute erzeugen
Beim Aktivieren dieser Option werden alle obligatorischen und optionalen Attribute, die im Schema definiert sind, generiert.

Nicht obligatorische Elemente erzeugen
Beim Aktivieren dieser Option werden alle obligatorischen und optionalen Elemente, die im Schema definiert sind, generiert.

Erste Wahl von obligatorischer Auswahl generieren
Beim Aktivieren dieser Option wird die erste Wahl aus einer obligatorischen Auswahl generiert/eingefügt.

Erzeuge "1" Element(e), wenn als wiederholend definiert (Schema/DTD)
Beim Aktivieren dieser Option wird die im Textfeld eingegebene Anzahl von sich wiederholenden Elementen generiert.

Elemente und Attribute mit Daten befüllen
Beim Aktivieren dieser Option werden die Datentype-Deskriptoren/Werte der jeweiligen Elemente/Attribute eingefügt. Z.B.: `Boolean = 1`, `xsd:string = string`, Max/Min inklusive = der im Schema definierte Wert.

13.5.12 Programmcode generieren...

Mit diesem Menüpunkt werden automatisch Java-, C++- oder C#-Klassendateien aus Definitionen im aktiven Schema (DTD oder XML-Schema) generiert.
Weitere Informationen dazu erhalten Sie in der Code Generator Dokumentation.

13.5.13 Cache-Speicher leeren

Mit diesem Befehl werden alle zwischengespeicherten Schema- und DTD-Dokumente aus dem Speicher gelöscht. Um Validierung und intelligentes Editieren zu beschleunigen, werden in XMLSpy 2005 alle zuletzt verwendeten Schemadokumente und externen geparsten Entities im Zwischenspeicher gespeichert. Informationen aus diesen Dokumenten werden auch mittels des Befehls Gehe zu Definition angezeigt.

Verwenden Sie diesen Befehl, wenn Sie nicht genügend Speicher in Ihrem System haben, oder wenn Sie zuletzt mit sehr vielen Dokumenten gearbeitet haben, die auf verschiedenen Schemas basieren.

© 2005 Altova GmbH

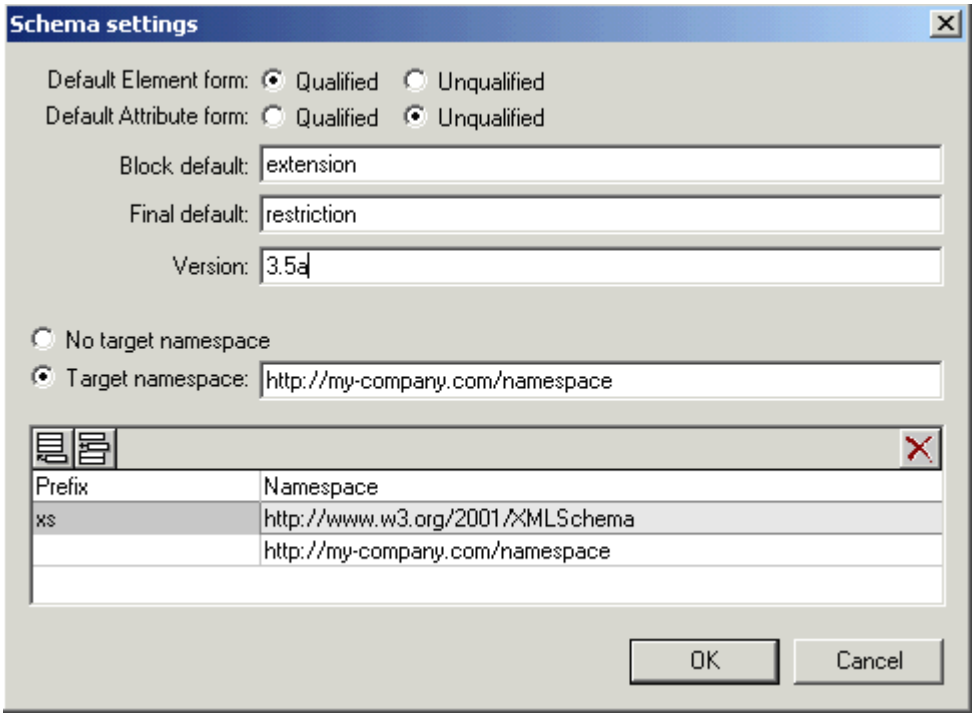
13.6 Menü "Schema-Design"

Im Schema-Design-Menü können Sie die Schema-Design-Ansicht von XMLSpy 2005 konfigurieren. In dieser Ansicht können Sie XML-Schemas über eine grafische Benutzeroberfläche erstellen. Das Menü steht dann zur Verfügung, wenn ein XML-Schemadokument in der Schema/WSDL-Ansicht aktiv ist.

Die Befehle in diesem Menü werden in diesem Abschnitt beschrieben.

13.6.1 Schema-Einstellungen

Mit diesem Befehl definieren Sie globale Einstellungen für das aktive Schema.
Dabei handelt es sich um Attribute und deren Werte für das XML-Schema-Dokument-Element `xs:schema`.



Sie können die Attribute `elementFormDefault` und `attributeFormDefault` des xs:schema-Elements jeweils auf `qualified` oder `unqualified` setzen. Weiters können Sie die Attribute `blockDefault`, `finalDefault` and `version` durch Eingabe der benötigten Werte in das jeweilige Textfeld definieren. Keines dieser fünf Attribute ist obligatorisch.

Um einen Target Namespace für das XML-Instanzdokument zu definieren, aktivieren Sie das Target Namespace-Optionsfeld und geben Sie den Target Namespace in das Textfeld ein. Bei Deklaration eines Target Namespace müssen Sie diesen Namespace im Schema-Dokument durch Eingabe einer Namespace-Zeile im unteren Bereich definieren. Sie können für diesen Namespace ein Präfix definieren oder ihn als Default Namespace verwenden (indem Sie das Präfix-Feld wie in der obigen Abbildung gezeigt leer lassen).

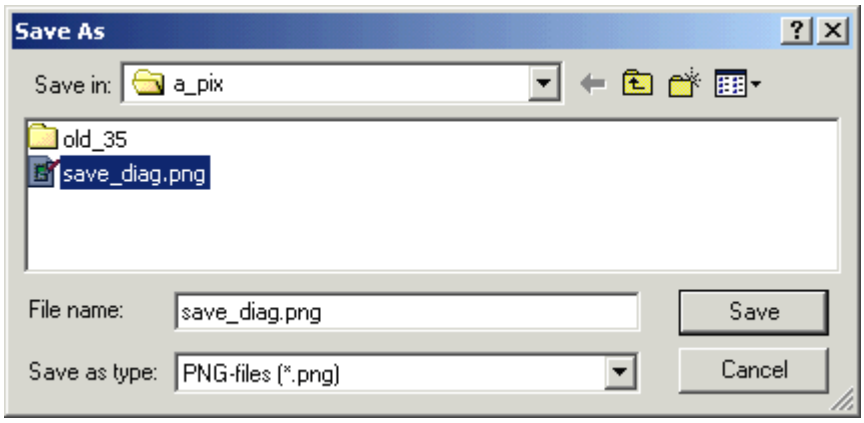
In der Textansicht sehen die obigen Schema-Einstellungen etwa folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.xmlspy.com/schemas/Altova/orgc
hart" xmlns="http://www.xmlspy.com/schemas/Altova/orgchart"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="unqualified"
attributeFormDefault="unqualified" blockDefault="extension"
finalDefault="restriction" version="3.5a">
    <xsd:notation name="Altova-Orgchart"
public="http://www.xmlspy.com/schemas/Altova/orgchart"/>
    <xsd:complexType name="DivisionType">
        <xsd:sequence>
            <xsd:element name="Name" type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>Division Name
```

Hinweis: Diese Einstellungen gelten nur für das aktive Dokument.

13.6.2 Diagramm speichern...

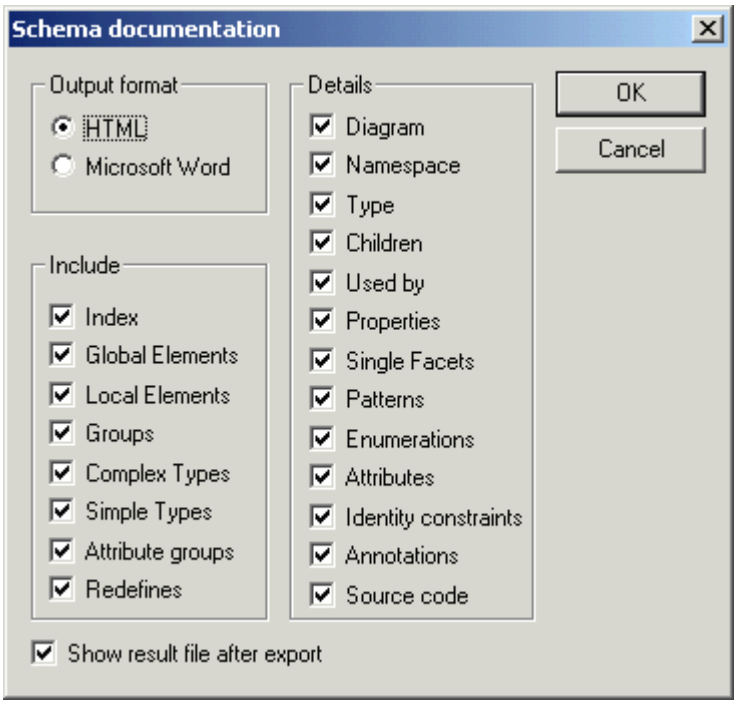
Mit diesem Befehl wird das Diagramm des im Hauptfenster angezeigten Content Model im PNG-Format unter dem gewünschten Pfad gespeichert.



13.6.3 Schema-Dokumentation generieren

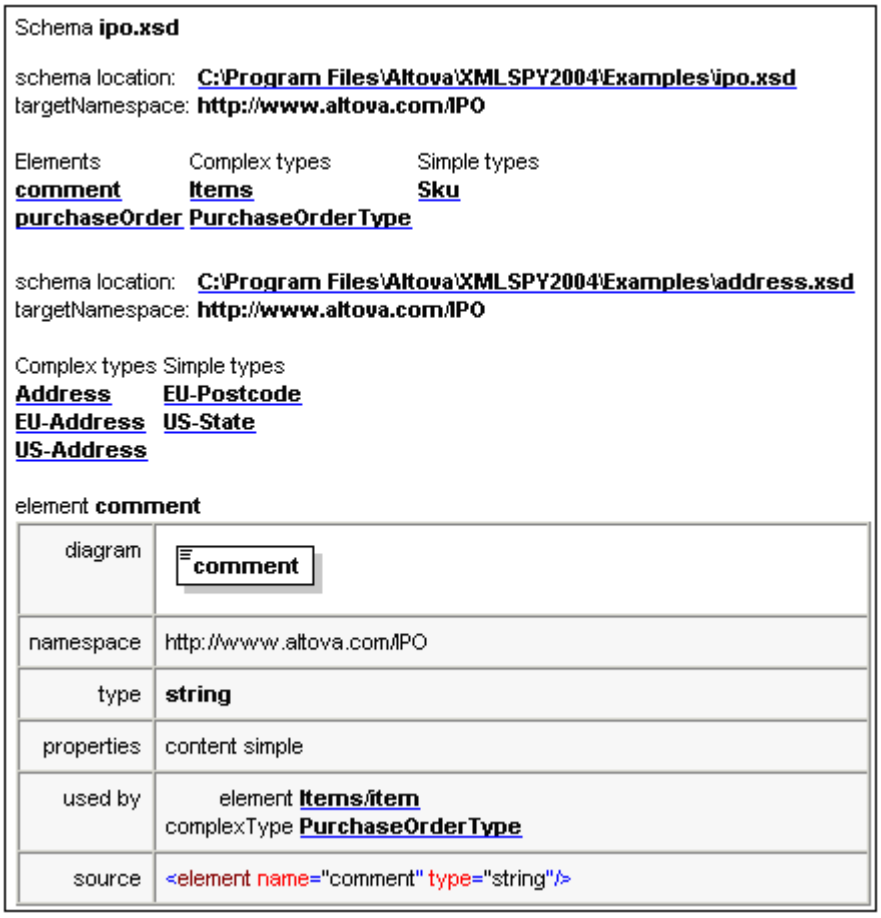
Mit diesem Befehl wird eine detaillierte Dokumentation zum Schema im HTML- oder MS Word-Format erzeugt. Die Dokumentation wird für diejenigen Komponenten generiert, die Sie im Dialogfeld "Schema-Dokumentation" auswählen (Aufruf des Dialogfelds mit dem Befehl "Schema-Dokumentation generieren", *siehe Abbildung unten*). Verwandte Elemente (Child-Elemente, ComplexTypes, usw.) werden in der Bildschirmausgabe mit Hyperlinks versehen, sodass Sie zwischen den einzelnen Komponenten navigieren können. Komponenten mit einem Content Model weisen auch Links zur jeweiligen Content Model-Definition auf. Beachten Sie, dass Schema-Dokumentation auch für **inkludierte Schemakomponenten** generiert wird.

© 2005 Altova GmbH



Wenn Sie auf den Befehl "Schema-Dokumentation generieren" klicken, erscheint

- der Bereich **Inkludieren,** in dem Sie auswählen können, welche Datenelemente in der Dokumentation inkludiert werden sollen. Bei Auswahl der Option "Index" werden alle verwandten Schemas mit ihren globalen Komponenten nach Komponententyp geordnet am Anfang der Datei aufgelistet.
- der Bereich **Details,** in dem die Einzelheiten aufgelistet werden, die für die einzelnen Komponenten inkludiert werden können. Wählen Sie die gewünschten Informationen aus. Bei Auswahl der Option "Diagramm" enthält das Dokument eine grafische Darstellung der einzelnen Schema-Datenelemente.
- Wenn Sie die Option "Resultat nach dem Export anzeigen" aktivieren, wird die erzeugte HTML-Datei in der Browser-Ansicht und eine erzeugte Word-Datei in MS Word angezeigt.



Schema **ipo.xsd**

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\ipo.xsd**
targetNamespace: **http://www.altova.com/IPO**

Elements	Complex types	Simple types
comment	**Items**	**Sku**
purchaseOrder	**PurchaseOrderType**	

schema location: **C:\Program Files\Altova\XMLSPY2004\Examples\address.xsd**
targetNamespace: **http://www.altova.com/IPO**

Complex types	Simple types
Address	**EU-Postcode**
EU-Address	**US-State**
US-Address	

element **comment**

diagram	comment
namespace	http://www.altova.com/IPO
type	**string**
properties	content simple
used by	element **Items/item** complexType **PurchaseOrderType**
source	<element name="comment" type="string"/>

In der oben gezeigten Abbildung sehen Sie die generierte Schema-Dokumentation mit einem Index (alle verwandten Schemas mit ihren globalen Komponenten nach Komponententyp geordnet) am Beginn des Dokuments.

13.6.4 Ansicht konfigurieren

Mit diesem Befehl können Sie die Anzeige des Content Model konfigurieren. Wenn Sie auf den Befehl klicken, wird rechts unten im XMLSpy 2005-Fenster das Dialogfeld "Schema-Ansicht konfigurieren" geöffnet. Hier sehen Sie bereits bei der Eingabe in das Dialogfeld, wie sich Ihre Einstellungen auswirken. Sobald Sie auf OK klicken, werden die Einstellungen aktiv und für das Content Model aller XML-Schemadateien, die in der Folge geöffnet werden, übernommen. Diese Einstellungen werden auch für die erzeugte Schema-Dokumentation und die Druckausgabe übernommen.

Definieren von Eigenschaftsdeskriptorzeilen für das Content Model
Sie können festlegen, welche Element- und Attributeigenschaften in der Content Model-Ansicht angezeigt werden sollen. Diese Eigenschaften werden als Grid-Zeilen in Komponentenkästchen angezeigt. Um Eigenschaftsdeskriptorzeilen zu definieren, gehen Sie vor wie folgt:

1. Klicken Sie auf dem Element- oder Attribut-Register auf das Symbol "Anhängen" oder "Einfügen" , um eine Eigenschaftsdeskriptorzeile hinzuzufügen. Die Zeile wird im Dialogfeld und zu den Elementkästchen in der Content Model-Ansicht hinzugefügt.
2. Wählen Sie aus der Dropdown-Liste die gewünschte Eigenschaft aus. *Siehe Abbildung*

 © 2005 Altova GmbH

unten.

3. Wiederholen Sie Schritt 1 und 2 für alle benötigten Eigenschaften.

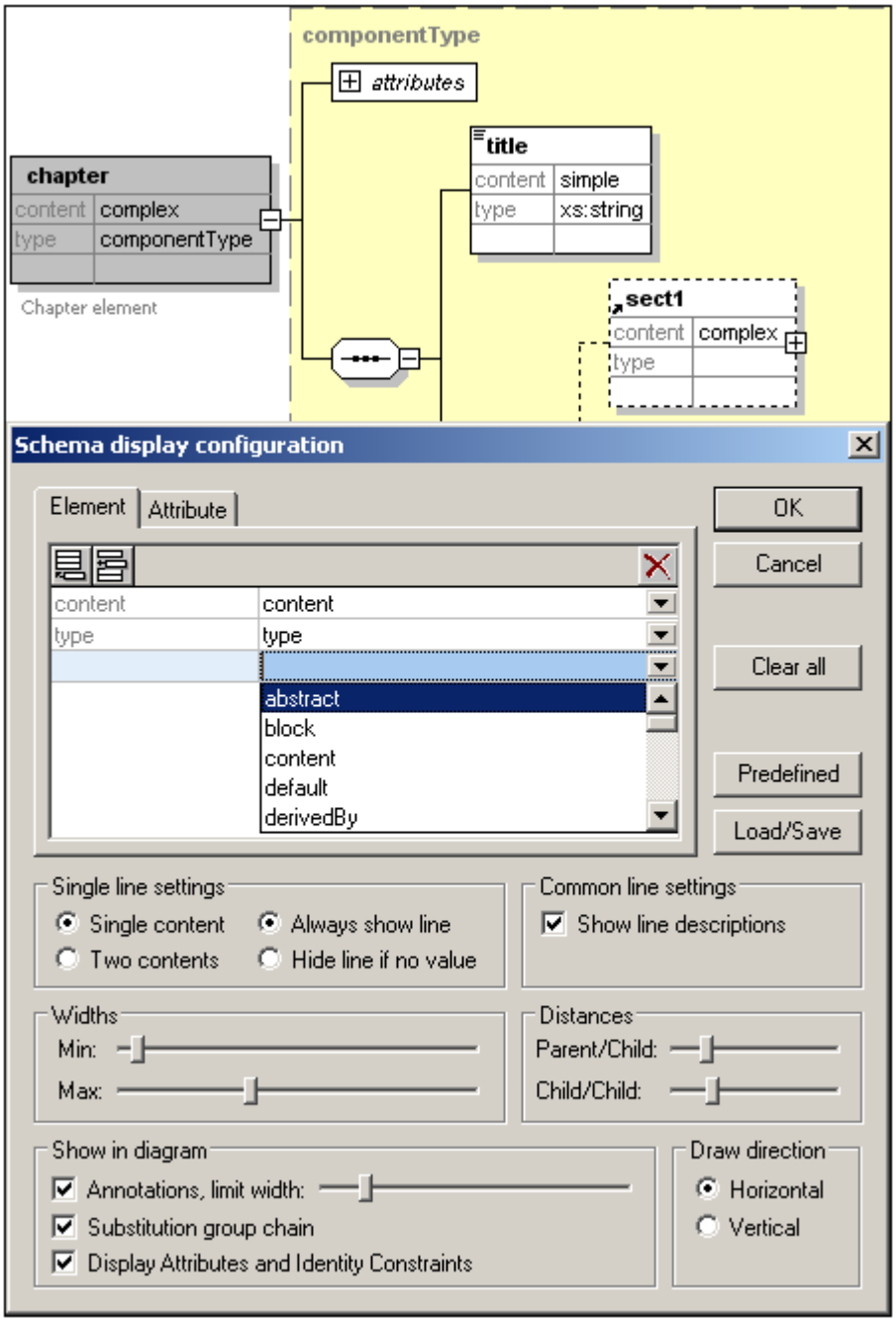
Die Content Model-Ansicht wird aktualisiert und zeigt die definierten Eigenschaftsdeskriptorzeilen für alle Elemente an, für die sie existieren.

Hinweis:

- Bei Attributen wird die von Ihnen definierte Konfiguration nur angezeigt, wenn Attribute (im Gegensatz zur Anzeige der Attribute in einem Bereich unterhalb der Content Model-Ansicht) im Diagramm angezeigt werden.
- Die konfigurierte Ansicht wird auf alle Content Model-Ansichten angewandt, die nach Definition der Ansicht, geöffnet werden.

Löschen einer Eigenschaftsdeskriptorzeile aus der Content Model-Ansicht
Um einzelne Eigenschaftsdeskriptorzeilen im Dialogfeld "Schema-Ansicht konfigurieren" zu löschen, wählen Sie die betreffende Zeile aus und klicken Sie auf das Symbol "Löschen" .

Einstellungen zur Konfiguration der Content Model-Ansicht
Die Content Model-Ansicht kann mit Hilfe von Einstellungen im Dialogfeld " Schema-Ansicht konfigurieren" konfiguriert werden. Oben wurde beschrieben, wie Sie definieren, welche Eigenschaftsdeskriptorzeilen in der Content Model-Ansicht angezeigt werden. Im Folgenden werden die anderen Einstellungen beschrieben:



Zeileneinstellungen (gilt pro Zeile)
Sie können festlegen, ob eine Eigenschaftsdeskriptorzeile Einzel- oder Doppel-Inhalte enthalten soll und ob Zeilen für jedes einzelne Element oder nur für Elemente, die diese Eigenschaft enthalten, angezeigt werden sollen. Aktivieren Sie das entsprechende Optionsfeld für die gewünschte Einstellung. Beachten Sie, dass diese beiden Einstellungen separat für einzelne Zeilen definiert werden können (markieren Sie die gewünschte Zeile und legen Sie die

 © 2005 Altova GmbH

Einstellung fest).

Allgemeine Zeileneinstellungen
Mit dieser Option blenden Sie die Zeilenbeschreibungen (d.h. den Namen der Eigenschaft) ein bzw. aus.

Breite
Mit Hilfe dieser Schieberegler können Sie die Mindest- und Maximalgröße der Elementrechtecke in der Content Model-Ansicht einstellen. Ändern Sie die Größe, wenn der beschreibende Text nicht zur Gänze sichtbar ist oder wenn Sie die Anzeige standardisieren möchten.

Abstände
Mit den Schiebern können Sie den horizontalen und vertikalen Abstand zwischen den Elemente am Bildschirm verändern.

Im Diagramm zeigen
Das Kontrollkästchen Annotationen erlaubt es, den Annotationstext ein- oder auszublenden und die Textbreite des Annotationstextes zu verändern. Weiters können Sie die Anzeige der Substitutionsgruppen ein- oder ausblenden. Wenn Sie dieses Kontrollkästchen aktivieren, werden die Attribute und Identity Constraints im Content Model-Diagramm angezeigt; andernfalls werden Sie am unterern Rand des Content Model-Fensters als Register angezeigt.

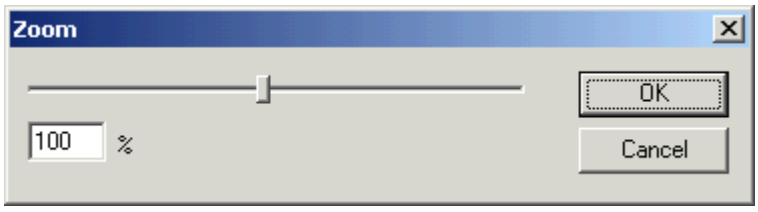
Zeichenrichtung
Diese Optionen definieren die Orientierung des Elementdiagramms am Bildschirm. Wählen Sie zwischen horizontal und vertikal.

Die Schaltfläche **Laden/Speichern** lädt und speichert die von Ihnen getätigten Einstellungen.
Die Schaltfläche **Vordefiniert** setzt die Konfiguration der Ansicht auf die Werkseinstellungen zurück.
Die Schaltfläche **Zurücksetzen** löscht alle Listenfeld-Einträge.

Editieren des Content Model im Diagramm selbst
Sie können Elementparameter- und Werte direkt im Inhaltsmodell ändern. Doppelklicken Sie auf die Eigenschaft, die Sie ändern wollen, und geben Sie einen neuen Wert ein. Wenn mehrere Werte zur Auswahl stehen, öffnet sich das Drop-down Listenfeld, ansonsten geben Sie den gewünschten Wert ein und bestätigen Sie mit der Eingabetaste.

13.6.5 Vergrößern/Verkleinern

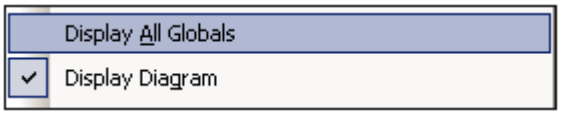
Mit diesem Befehl stellen Sie den Zoomfaktor der Ansicht des Content Model ein. Diese Funktion eignet sich zum Verkleinern eines großen Content Model, sodass das gesamte Content Model im Hauptfenster zu sehen ist. Der Zoomfaktor beträgt zwischen 10 % und 200 % der tatsächlichen Größe.



Ziehen Sie den Schieber um den Zoomfaktor zu verändern, oder geben Sie im Textfeld den gewünschten Zoomfaktor ein.

13.6.6 Globale Elemente anzeigen

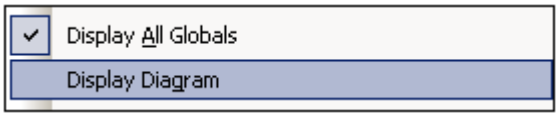
Mit diesem Befehl wechseln Sie von der Content Model-Ansicht zur Schemaübersicht, um alle globalen Komponenten im Schema anzuzeigen. Es ist entweder dieser Befehl oder der Befehl "Diagramm anzeigen" aktiv. Der jeweils aktive Befehl wird durch ein Häkchen auf der linken Seite angezeigt. (*siehe Abbildung unten).*



Alternativ dazu können Sie auch auf das Symbol **Globale Elemente anzeigen** am oberen Rand des Content Model klicken, um zur Schema-Übersicht zurückzukehren.

13.6.7 Diagramm anzeigen

Mit diesem Befehl zeigen Sie die Content Model-Ansicht der ausgewählten globalen Komponente an - falls die ausgewählte globale Komponente ein Content Model hat. Globale Komponenten mit einem Content Model (ComplexTypes, Elemente und Element-Gruppen) werden mit dem Symbol links davon gekennzeichnet. Es ist entweder dieser Befehl oder der Befehl "Globale Elemente anzeigen" aktiv. Der jeweils aktive Befehl wird durch ein Häkchen auf der linken Seite angezeigt. (*siehe Abbildung unten).*



Alternativ dazu haben Sie folgende Möglichkeiten, um zur Content Model-Ansicht zu wechseln:

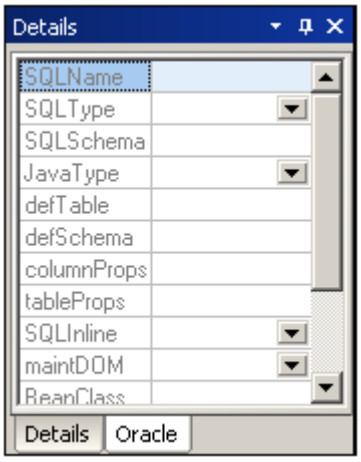
- Klicken Sie auf das Symbol neben der Komponente, die Sie anzeigen möchten, oder
- Doppelklicken Sie auf den Namen der Komponente im Komponenten-Navigator (oberste rechte Eingabehilfe).

13.6.8 Oracle Schema Extensions aktivieren

XMLSpy 2005 unterstützt Oracle Schema Extensions zur Verwendung in Oracle 9i Project XDB. Mit Hilfe dieser Extensions können Sie definieren und konfigurieren, wie Oracle 9i Project XDB XML-Dokumente intern speichert. Diese XML-Dokumente können dann über SQL Queries und Legacy Tools aufgerufen werden. Nähere Informationen dazu finden Sie auf der Oracle Website.

Bei Auswahl des Befehls **Oracle Schema Extensions aktivieren** geschieht Folgendes:

- Der XDB Namespace wird am `schema` Element deklariert: `xmlns:xdb="http://xmlns.oracle.com/xdb"`.
- In der Eingabehilfe "Details" wird ein Register "Oracle" angelegt, über das Sie Attribute - einschließlich XDB-spezifischer Attribute - zu Schema-Elementen wie `xsd:complexType` and `xsd:element` hinzufügen können.

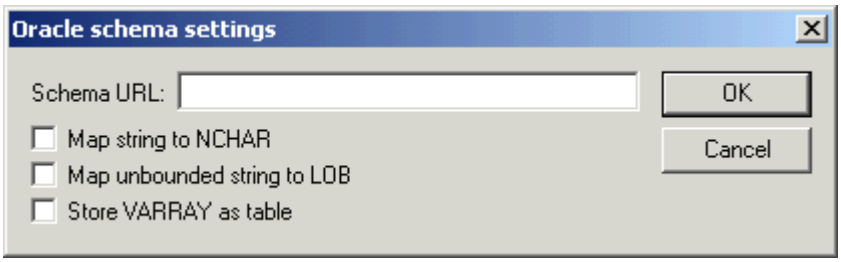


Oracle Extensions können für complexType, Elemente und Attribute definiert werden. Verwenden Sie die Eingabehilfe genau wie in XMLSpy 2005.

Hinweis: Sie können diesen Menübefehl aktivieren oder deaktivieren, d.h. Sie können Extensions ein- oder ausschalten. Bei aktivierten Oracle Extensions wird links neben dem Befehl ein Häkchen angezeigt. Bei Deaktivierung von Oracle Extensions (durch Klicken auf den aktivierten Befehl) werden die XDB Namespace Deklaration und alle XDB Extensions in der Datei gelöscht. Sie werden durch eine Meldung darauf hingewiesen, dass diese Informationen unwiderruflich gelöscht werden!

13.6.9 Oracle Schema-Einstellungen

Mit diesem Befehl können Sie globale Einstellungen für Oracle Schemaerweiterungen vornehmen.



Diese Menüoption kann nur aufgerufen werden, wenn Sie zuvor Oracle Schema Extensions aktiviert haben (Menüoption "Oracle Schema Extensions aktivieren").

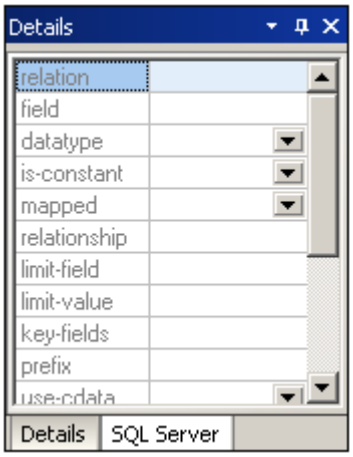
13.6.10 Microsoft SQL Server Schema Extensions aktivieren

XMLSpy 2005 unterstützt Microsoft SQL Server 2000 Schema Extensions zur Verwendung mit Microsoft SQL Server. Durch Verwendung dieser Extensions können Sie definieren und konfigurieren, wie Microsoft SQL Server Ihre XML-Dokumente intern speichert. Diese XML-Dokumente können dann über SQL-Abfragen und Legacy Tools aufgerufen werden. Weitere Informationen zu diesem Thema finden Sie auf der Microsoft Website.

Bei Auswahl dieser Menüoption geschieht Folgendes:

- Der SQL Server Namespace wird am `schema` Element deklariert: `xmlns:sql="urn:schemas-microsoft-com:mapping-schema".`
- In der Eingabehilfe "Details" wird ein Register "SQL Server" angelegt, über das Sie

Attribute zu Schema-Elementen wie `xsd:element` hinzufügen können.



Wenn SQL Server Extensions für eine Schema-Komponente definiert werden können, steht das Register "SQL Server" in der Eingabelhilfe "Details" zur Verfügung, wenn die Komponente ausgewählt ist. Verwenden Sie die Eingabehilfe genau wie in XMLSpy 2005.

Hinweis: Sie können diesen Menübefehl aktivieren oder deaktivieren, d.h. Sie können Extensions ein- oder ausschalten. Bei aktivierten SQL Server Extensions wird links neben dem Befehl ein Häkchen angezeigt. Bei Deaktivierung von SQL Server Extensions (durch Klicken auf den aktivierten Befehl) werden die SQL Server Namespace Deklaration und alle SQL Extensions in der Datei gelöscht. Sie werden durch eine Meldung darauf hingewiesen, dass diese Informationen unwiderruflich gelöscht werden!

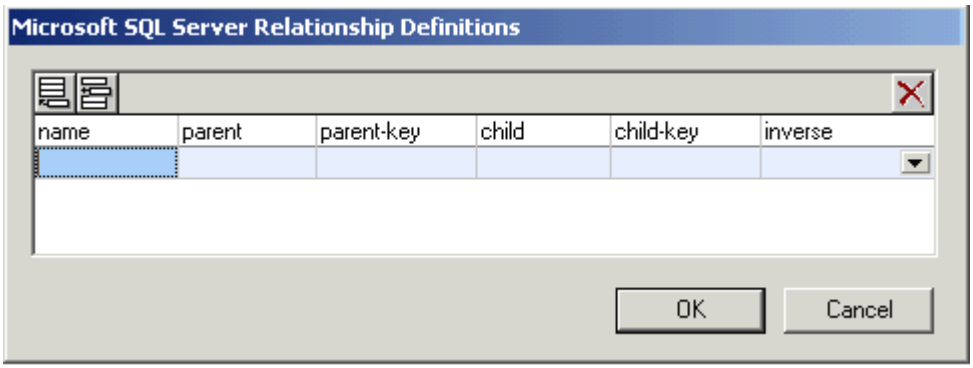
13.6.11 Named Schema Relationships

Mit diesem Befehl können Sie "named relationships" definieren, die benötigt werden, um das Erstellen von Dokumenthierarchien zu erlauben.
Diese Menüoption kann nur aufgerufen werden, wenn Sie zuvor SQL Server Schema Extensions aktiviert haben (Menüoption "SQL Server Schema Extensions aktivieren").

So erstellen Sie Named Schema Relationships:

1. Klicken Sie auf das Symbol "Einfügen" oder "Anhängen" , um im Dialogfeld eine neue Zeile einzufügen.
2. Klicken Sie in das neue Feld, um den Namen der Beziehung einzugeben.
3. Mit OK bestätigen Sie Ihre Eingaben.

 © 2005 Altova GmbH



Es wird ein SQL Relationship Element erzeugt, das unter der Namespace Deklaration eingefügt wird.

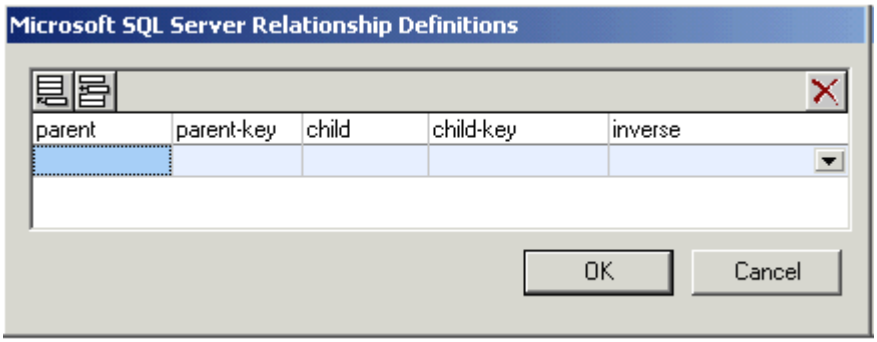
Hinweis: Klicken Sie auf das Symbol "Löschen" um eine Zeile aus dem Dialogfeld zu löschen.

13.6.12 Unnamed Element Relationships

Mit diesem Befehl können Sie Unnamed Relationships definieren, um Informationen für das Erstellen von Dokumenthierarchien bereitstellen zu können. Diese Menüoption kann nur aufgerufen werden, wenn Sie zuvor SQL Server Schema Extensions aktiviert haben (Menüoption "SQL Server Schema Extensions aktivieren").

So erstellen Sie Unnamed Schema Relationships:

1. Klicken Sie auf das Symbol "Einfügen" oder "Anhängen" , um im Dialogfeld eine neue Zeile einzufügen.
2. Klicken Sie in das Feld, um den Namen der Beziehung einzugeben.
3. Mit OK bestätigen Sie Ihre Eingaben.



Hierdurch wird ein SQL Relationship Element für das ausgewählte Schemaelement erzeugt.

Hinweis: Klicken Sie auf das Symbol "Löschen" um eine Zeile aus dem Dialogfeld zu löschen.

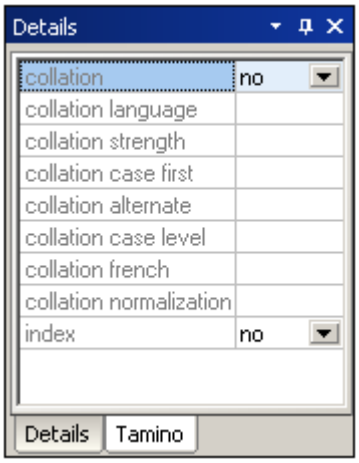
13.6.13 Tamino Schema Extensions aktivieren

XMLSpy 2005 unterstützt Tamino Schema Extensions zur Verwendung mit Tamino Server. Software AG und die Altova GmbH haben ein weltweit gültiges Abkommen unterzeichnet, mit dem Benutzer von XMLSpy 2005 einfachen und kostengünstigen Zugang zu Tamino (über ein Tamino Datenbankbündel) erhalten. Weitere Informationen erhalten Sie im Abschnitt Tamino.

Durch Verwendung der Tamino Schema Extensions können Sie definieren, wie Tamino Ihre XML-Dokumente intern speichert. Diese XML-Dokumente können dann über Abfragetools und Legacy Tools aufgerufen werden.

Bei Auswahl dieser Menüoption geschieht Folgendes:

- Wenn das XML-Schema Namespace Präfix nicht `xs:` ist, werden Sie von XMLSpy 2005 gefragt, ob das Präfix in das von Tamino benötigte Präfix in `xs:` geändert werden darf.
- Wenn die Dateierweiterung nicht `.tsd` ist, werden Sie von XMLSpy 2005 gefragt, ob die Dateierweiterung in die von Tamino benötigte Erweiterung `.tsd`. geändert werden darf.
- Es erscheint das Dialogfeld "Tamino-Eigenschaften", in dem Sie die Tamino-Eigenschaften einstellen können. Nähere Informationen dazu siehe Tamino-Eigenschaften.
- Der Tamino Namespace wird am `schema` Element: `xmlns:tsd="http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition"` deklariert.
- In der Eingabehilfe "Details" wird ein Register "Tamino" angelegt, über das Sie Attribute zu Schema-Elementen wie `xsd:element` hinzufügen können.



Wenn Tamino Extensions für eine Schema-Komponente definiert werden können, steht das Register "Tamino" in der Eingabelhilfe "Details" zur Verfügung, wenn die Komponente ausgewählt ist. Verwenden Sie die Eingabehilfe genau wie in XMLSpy 2005.

Hinweis: Sie können diesen Menübefehl aktivieren oder deaktivieren, d.h. Sie können Extensions ein- oder ausschalten. Bei aktivierten Tamino Extensions wird links neben dem Befehl ein Häkchen angezeigt. Bei Deaktivierung von Tamino Extensions (durch Klicken auf den aktivierten Befehl) werden die Tamino Namespace Deklaration und alle Tamino Extensions in der Datei gelöscht. Sie werden durch eine Meldung darauf hingewiesen, dass diese Informationen unwiderruflich gelöscht werden!

13.6.14 Tamino Schema-Eigenschaften

Mit diesem Befehl öffnen Sie das Dialogfeld "Tamino-Eigenschaften". Hier erhalten Sie Informationen zur Tamino Datenbank und Schemaeigenschaften. Nähere Informationen zu diesen Eigenschaften finden Sie im Abschnitt Tamino dieser Dokumentation unter Tamino Schema-Eigenschaften.

© 2005 Altova GmbH

13.6.15 Verbindung zu SchemaAgent Server herstellen

Mit diesem Befehl können Sie über die IP-Adresse des Servers eine Verbindung zu einem SchemaAgent Server herstellen. Nähere Informationen dazu finden Sie im Abschnitt Herstellen einer Verbindung mit SchemaAgent Server im Abschnitt "Schema/WSDL-Ansicht" dieses Benutzerhandbuchs. Nach Herstellung der Verbindung zum SchemaAgent Server erscheint oben in der Hauptfensterleiste eine Meldung mit Verbindungsinformationen. Sie haben nun unbeschränkten Zugriff auf alle Schemas und Schema-Konstruktoren im Server-Suchpfad.

Hinweis: Damit eine Verbindung zustande kommt, müssen Sie den Altova SchemaAgent Client auf Ihrem Rechner installiert haben und eine gültige Lizenz dafür besitzen.

13.6.16 Verbindung mit SchemaAgent Server trennen

Dieser Befehl steht zur Verfügung, wenn eine Verbindung zu einem SchemaAgent Server zustandegekommen ist. Bei Auswahl dieses Befehls wird die Verbindung zwischen XMLSpy 2005 und dem SchemaAgent Server getrennt.

13.6.17 In SchemaAgent anzeigen

Wenn Sie den Mauszeiger über den Menübefehl **In SchemaAgent anzeigen** halten, erscheint ein Untermenü das Optionen enthält, welche Schemas in SchemaAgent angezeigt werden sollen. Nähere Informationen zu diesen Optionen finden Sie in dieser Dokumentation im Abschnitt "Schema/WSDL-Ansicht" unter Anzeigen von Schemas in SchemaAgent.

13.6.18 Erweiterte Validierung

Mit diesem Befehl können Sie das aktuell aktive Schema und die damit verwandten Schemas validieren. Eine Beschreibung zu dieser Funktion finden Sie in dieser Dokumentation im Abschnitt "Schema/WSDL-Ansicht" unter Erweiterte Validierung.

13.7 Menü "XSL/XQuery"

Mit Hilfe der XSL Transformationssprache können Sie festlegen, wie ein XML-Dokument in andere XML-Dokumente oder Textdateien konvertiert werden soll. Eine Art von XML-Dokument, die mittels eines XSLT-Dokuments generiert wird, ist ein FO-Dokument, das anschließend weiter in ein PDF-Ausgabedokument konvertiert wird.
XMLSpy 2005 enthält integrierte XSLT-Prozessoren (für XSLT 1.0 und XSLT 2.0) und kann mit einem FO-Prozessor auf Ihrem System verknüpft werden, um XML-Dateien zu transformieren und verschiedene Arten von Ausgabedokumenten zu generieren. Der Pfad zum FO-Prozessor muss im Dialogfeld "Optionen" auf dem Register XSL (Extras | Optionen) angegeben werden, um direkt innerhalb der XMLSpy 2005 Benutzeroberfläche verwendet werden zu können.

Des Weiteren verfügt XMLSpy 2005 über einen integrierten XQuery-Prozessor, mit Hilfe dessen Sie XQuery-Dokumente (mit oder ohne Referenz auf ein XML-Dokument) ausführen können.

Die Befehle für die oben beschriebenen Transformationen finden Sie im Menü **XSL/XQuery**. Zusätzlich dazu enthält dieses Menü auch Befehle für die Arbeit mit dem Altova XSLT/XQuery Debugger.

Technische Informationen zum integrierten XSLT- und XQuery-Prozessor finden Sie unter Altova XQuery-Prozessor, Altova XSLT 1.0 Prozessor und Altova XSLT 2.0 Prozessor.

13.7.1 XSL-Transformation

F10

Mit diesem Befehl führen Sie mittels eines zugewiesenen XML-Dokuments eine **XSL-Transformation** durch. Die Transformationen kann entweder mit Hilfe des entsprechenden Altova XSLT-Prozessors (Altova XSLT 1.0-Prozessor für XSLT 1.0 Stylesheets; Altova XSLT 2.0-Prozessor für XSLT 2.0 Stylesheets), mit Hilfe des MSXML-Moduls von Microsoft oder eines externen XSLT-Prozessors durchgeführt werden. Der in Verbindung mit diesem Befehl verwendete Prozessor, wird im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register XSL festgelegt.

Sollte Ihr XML-Dokument eine Referenz zu einem XSLT Stylesheet enthalten, wird dieses Stylesheet automatisch für die XSLT Transformation verwendet. (Ein XSLT Stylesheet kann über den Befehl XSL zuweisen zugewiesen werden), Alternativ können Sie im Dialogfeld "Projekteigenschaften" ein XSLT Stylesheet auch für einzelne Ordner definieren. Klicken Sie mit der rechten Maustaste auf den/die Projektordner oder die zu transformierenden Datei(en) und wählen Sie XSL-Transformation. Wurde einer XML-Datei kein XSLT Stylesheet zugewiesen, werden Sie aufgefordert das XSLT Stylesheet anzugeben.

Weitere Informationen finden Sie unter: Altova XSLT 1.0 Prozessor und Altova XSLT 2.0 Prozessor.

13.7.2 XSL:FO-Transformation

Strg+F10

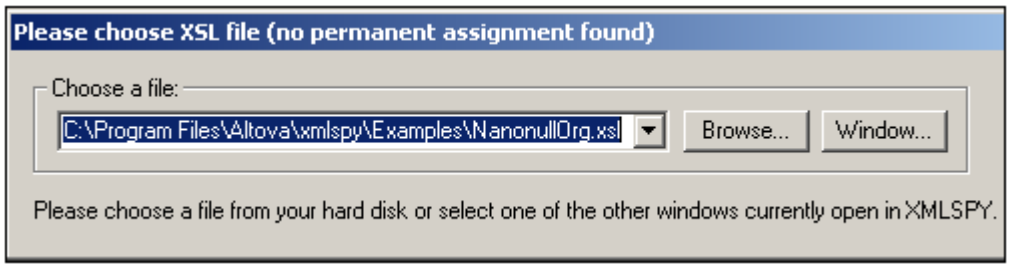
FO ist ein XML-Format zur Beschreibung von seitenweise formatierten Dokumenten. Ein FO-Prozessor wie z.B. der FOP-Prozessor von Apache XML Projects verarbeitet FO, um eine PDF-Datei zu erzeugen, die Generierung eines PDF-Dokuments erfolgt also in zwei Schritten.

© 2005 Altova GmbH

1. Das XML-Dokument wird mittels eines XSLT Stylesheet (auch als XSL-FO bezeichnet) in ein FO-Dokument transformiert.
2. Aus dem FO-Dokument wird ein PDF-Dokument (oder ein anderes Ausgabeformat) generiert.

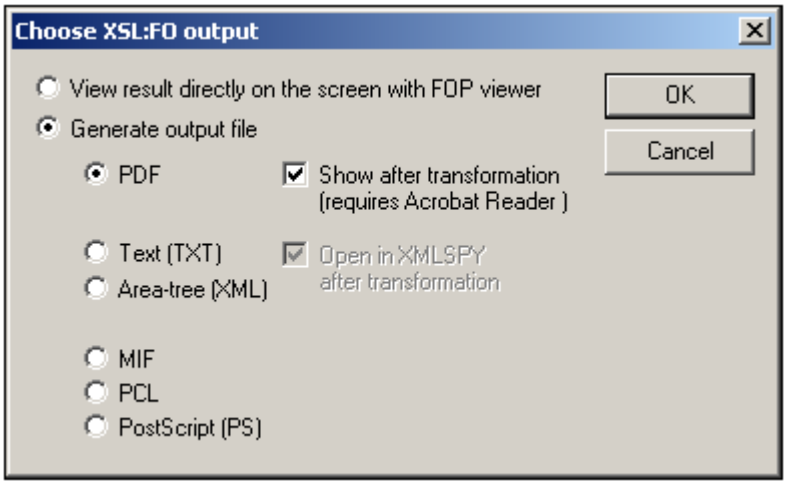
Mit dem Befehl **XSL:FO-Transformation** wird ein XML-Dokument oder ein FO-Dokument ins PDF-Format transformiert.

- Wird die **XSL:FO-Transformation** an einem XML-Quelldokument durchgeführt, werden beide oben beschriebenen Schritte nacheinander durchgeführt. Wenn das für die Transformation in FO benötigte XSLT (oder XSL-FO) Stylesheet im XML-Dokument nicht referenziert ist, werden Sie aufgefordert, eines für die Transformation zuzuweisen (*siehe Abbildung unten*).



Das erzeugte FO-Dokument wird von dem FO-Prozessor, der im Dialogfeld "Optionen" (**Extras | Optionen**), Register XSL definiert wurde, direkt verarbeitet.
Wird der Befehl **XSL:FO-Transformation** an einem FO-Dokument ausgeführt, wird das Dokument mit dem FO-Prozessor, der im Dialogfeld "Optionen" (**Extras | Optionen**), Register XSL definiert wurde, verarbeitet.

XSL:FO-Transformationsausgabe
Wenn Sie den Befehl **XSL:FO-Transformation** aufrufen, wird das Dialogfeld "XSL:FO-Ausgabe wählen" angezeigt (*siehe Abbildung unten*). (Wenn dem aktiven XML-Dokument kein XSLT Stylesheet zugewiesen wurde, werden Sie zuerst nach der XSLT-Datei gefragt).



Sie können das vom FO-Prozessor erzeugte Dokument mit Hilfe des FOP-Viewers direkt am Bildschirm anzeigen oder Sie können eine Ausgabedatei in einem der folgenden Formate erzeugen: PDF, Text, als XML-Baumdiagramm, MIF PCL oder PostScript.

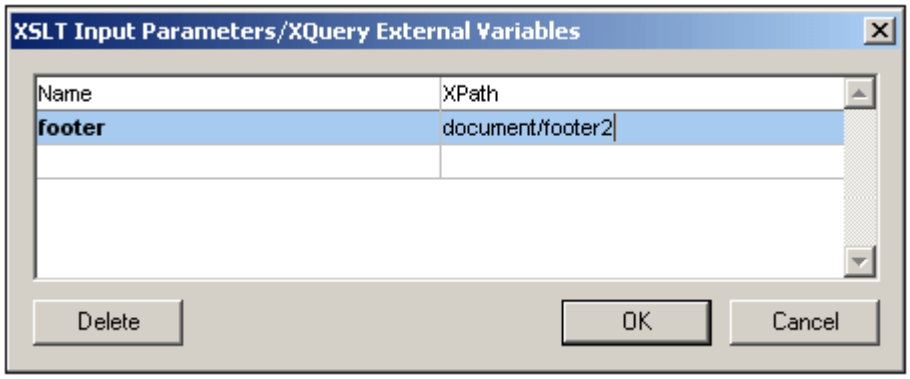
Hinweis:

- Der Apache FOP-Prozessor kann kostenlos über den Link im Altova Download Center heruntergeladen werden. Laden Sie FOP herunter und installieren Sie den Prozessor und geben Sie anschließend den Pfad zur FOP Batch-Datei im Dialogfeld "Optionen" (**Extras | Optionen**) auf dem Register XSL an.
- Sie können den Befehl XSL:FO-Transformation nicht nur an der aktiven Datei im Hauptfenster anwenden, sondern auch an jeder Datei/jedem beliebigen Ordner die/den Sie im Projektordner auswählen. Klicken Sie dazu mit der rechten Maustaste und wählen Sie den Befehl **XSL:FO-Transformation**. Es wird das XSLT Stylesheet verwendet, das dem ausgewählten Projektordner zugewiesen wurde.

13.7.3 XSL-Parameter/XQuery-Variablen

Mit dem Befehl **XSL-Parameter/XQuery-Variablen** öffnen Sie das Dialogfeld "XSLT-Eingabeparameter/externe XQuey-Variablen" (*siehe unten*). Sie können den Namen von einem oder mehreren Parametern oder den Namen von einer oder mehreren XQuery-Variablen und die dazugehörigen Werte angeben. Der/die Parameterwert(e) werden an den/die Parameter in der ausgewählten XSLT-Datei übergeben, wenn eine XML-Datei mit Hilfe des Befehls **XSL-Transformation** aus dem XSL-Menü verarbeitet wird. Wenn Sie den Befehl **XQuery-Ausführung** auswählen, werden externe XQuery-Variablenwerte an das XQuery-Dokument übergeben.

Verwendung von XSLT-Parametern
Bei dem Wert, den Sie für den Parameter eingeben, kann es sich um einen XPath ohne Anführungszeichen oder einen von Anführungszeichen eingeschlossenen Textstring handeln.



Hinweis: Sobald Sie im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" die Parameterwerte eingegeben haben, werden diese für alle zukünftigen Transformationen verwendet, bis sie ausdrücklich gelöscht werden oder die Applikation neu gestartet wird. Parameter, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden auf Applikationsebene definiert und werden von diesem Punkt an bei jeder Transformation, die über die IDE durchgeführt wird, an das jeweilige XSLT-Dokument übergeben. Dies bedeutet:

- Parameter sind nicht mit einem bestimmten Dokument verknüpft und
- Alle Parameter, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden gelöscht, sobald die Applikation (XMLSpy 2005) geschlossen wird.

Verwendung von XSLT-Parametern

© 2005 Altova GmbH

Im folgenden Beispiel wählen wir die für das PDF-Dokument benötigte Fußzeile aus drei Wahlmöglichkeiten im XML-Dokument (`footer1`, `footer2`, `footer3`) aus.

```
<?xml version="1.0" encoding="UTF-8"?>
<document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="C:\workarea\footers\footers.xsd">
   <footer1>Footer 1</footer1>
   <footer2>Footer 2</footer2>
   <footer3>Footer 3</footer3>
   <title>Document Title</title>
   <para>Paragraph text.</para>
   <para>Paragraph text.</para>
</document>
```

Die XSLT-Datei enthält in der Vorlage für das Root-Element einen lokalen Parameter namens `footer`. Dieser Parameter hat einen Standardwert `footer1`. Der Parameter wird anschließend in der Vorlage in der Definition des Footer-Blocks mit einem Wert `$footer` instanziiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format">
   ...
   <xsl:template match="/">
      <xsl:param name="footer" select="document/footer1" />
      <fo:root>
         <xsl:copy-of select="$fo:layout-master-set" />
         <fo:page-sequence master-reference="default-page"
            initial-page-number="1" format="1">
            <fo:static-content flow-name="xsl-region-after"
               display-align="after">
               ...
               <fo:inline color="#800000" font-size="10pt" font-weight="bold">
                  <xsl:value-of select="$footer"/>
               </fo:inline>
               ...
            </fo:static-content>
         </fo:page-sequence>
      </fo:root>
   </xsl:template>
</xsl:stylesheet>
```

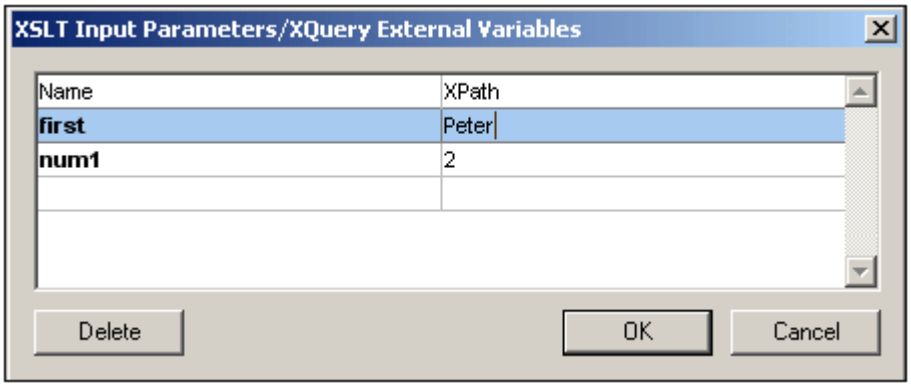
Sie können im XSLT-Eingabeparameter-Dialogfeld einen neuen Wert für den Parameter `footer` eingeben, z.B. den XPath: `document/footer2` (*siehe Abbildung oben)* oder einen Textstring. Während der Transformation wird dieser Wert an den Parameter `footer` in der Vorlage für das Root-Element übergeben und beim Instanziieren des Footer-Blocks verwendet.

Hinweis:

- Wenn Sie den Befehl **XSL:FO-Transformation** (**XSL/XQuery | XSL:FO-Transformation**) verwenden, werden Parameter, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben werden, **nicht** an das Stylesheet übergeben. Damit diese Parameter in der PDF-Ausgabe verwendet werden, führen Sie mittels des Befehls XSLT-Transformation **(XSL/XQuery | XSL-Transformation)** zuerst eine XSL-FO-Transformation durch und anschließend mit Hilfe des Befehls **XSL:FO-Transformation** (**XSL/XQuery | XSL:FO-Transformation**) eine FO-PDF-Transformation.
- Falls Sie einen anderen als den integrierten Altova XSLT-Prozessor verwenden, werden Parameter, die Sie mit Hilfe des Befehls "XSLT-Eingabeparameter" eingeben, nicht an den externen Prozessor übergeben.

Verwendung von externen XQuery-Variablen
Beim Wert, den Sie für eine externe XQuery-Variable eingeben, muss es sich um einen

Literalwert ohne Anführungszeichen handeln. Der Datentyp der externen Variable wird in der Variablendeklaration im XQuery-Dokument angegeben.



Hinweis: Sobald Sie im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eine Reihe von externen XQuery-Variablen eingegeben haben, werden diese für alle zukünftigen Transformationen verwendet, bis sie ausdrücklich gelöscht werden oder die Applikation neu gestartet wird. Variablen, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden auf Applikationsebene definiert und werden von diesem Punkt an bei jeder Transformation, die über die IDE durchgeführt wird, an das jeweilige XSLT-Dokument übergeben. Dies bedeutet:

- Variablen sind nicht mit einem bestimmten Dokument verknüpft und
- Alle Variablen, die im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurden, werden gelöscht, sobald die Applikation (XMLSpy 2005) geschlossen wird.

Anwendungsbeispiel für externe XQuery-Variablen
Im folgenden Beispiel wird im XQuery-Dokument eine Variable `$first` deklariert und anschließend in der Return Clause des FLWOR-Ausdrucks verwendet:

```
xquery version "1.0";
declare variable $first as xs:string external;
let $last := "Jones"
return concat($first, " ", $last )
```

Diese XQuery gibt `Peter Jones` zurück, wenn der Wert der externen Variable (der im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurde) `Peter` ist. Beachten Sie Folgendes:

- Das externe Schlüsselwort in der Variablendeklaration im XQuery-Dokument gibt an, dass es sich bei dieser Variable um eine externe Variable handelt.
- Die Definition des statischen Typs der Variable ist optional. Wenn der Datentyp für die Variable nicht in der Variablendeklaration angegeben wurde, wird dem Variablenwert der Typ `xdt:untypedAtomic` zugewiesen.
- Wenn im XQuery-Dokument eine externe Variable deklariert wurde, aber keine externe Variable dieses Namens an das XQuery-Dokument übergeben wird, wird eine Fehlermeldung ausgegeben.
- Wenn eine externe Variable deklariert und im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eingegeben wurde, wird diese als Gültigkeitsbereich für das auszuführende XQuery-Dokument verwendet. Wird eine neue Variable dieses Namens im XQuery-Dokument deklariert, setzt diese die externe Variable für den Gültigkeitsbereich vorübergehend außer Kraft. Das unten gezeigte XQuery-Dokument gibt z.B. `Paul Jones` zurück, obwohl die externe in-scope Variable `$first` den Wert `Peter` hat.

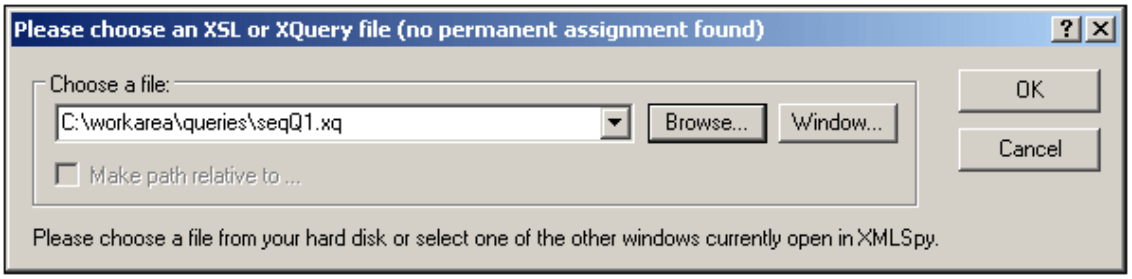
 © 2005 Altova GmbH

```
xquery version "1.0";
declare variable $first as xs:string external;
let $first := "Paul"
let $last := "Jones"
return concat($first, " ", $last )
```

Hinweis: Es handelt sich nicht um einen Fehler, wenn im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" eine externe XQuery-Variable (oder ein XSLT-Parameter) definiert ist, diese im XQuery-Dokument aber nicht verwendet wird. Auch wenn ein XSLT-Parameter (oder eine externe XQuery-Variable) im Dialogfeld "XSLT-Eingabeparameter/externe XQuery-Variablen" definiert ist, aber in einer XSLT-Transformation nicht verwendet wird, handelt es sich nicht um einen Fehler.

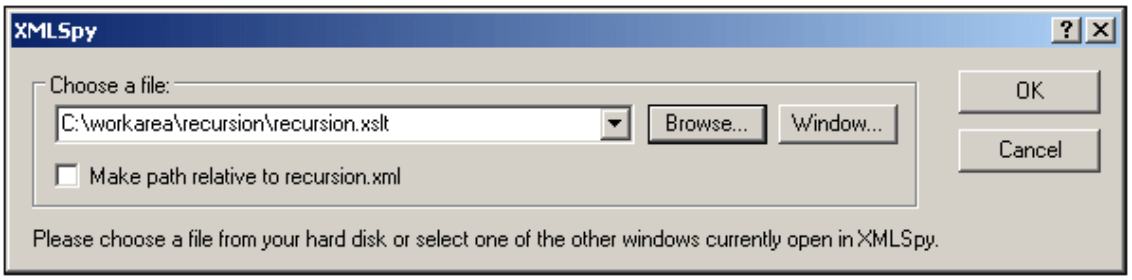
13.7.4 XQuery-Ausführung

Mit diesem Befehl wird ein XQuery-Dokument ausgeführt. Der Befehl steht zur Verfügung, wenn eine XQuery- oder XML-Datei aktiv ist. Bei Aufruf von einer XML-Datei aus erscheint ein Dialogfeld, in dem Sie aufgefordert werden, die XML-Datei mit einer XQuery-Datei zu verknüpfen.



13.7.5 XSL zuweisen...

Mit diesem Befehl weisen Sie einem XML-Dokument ein XSLT Stylesheet zu. Wenn Sie auf den Befehl klicken, wird ein Dialogfeld geöffnet, in dem Sie angeben können, welche XSLT-Datei zugewiesen werden soll.



Es wird eine `xml-stylesheet` Verarbeitungsanweisung in das XML-Dokument eingefügt:

```
<?xml-stylesheet type="text/xsl"
   href="C:\workarea\recursion\recursion.xslt"?>
```

Beachten Sie, dass Sie den Pfad zur zugewiesenen Datei durch Aktivieren des Kontrollkästchens "Pfad relativ machen zu..." relativ machen können.

13.7.6 XSL:FO zuweisen...

Mit diesem Befehl weisen Sie einem XML-Dokument ein XSL Stylesheet für die FO-Transformation zu. Der Befehl öffnet ein Dialogfeld, in dem Sie angeben können, welche XSL oder XSLT Datei zugewiesen werden soll. Die benötigte Verarbeitungsanweisung wird in das XML-Dokument eingefügt:

```
<?xmlspyxslfo C:\Program Files\Altova\xmlspy\Examples\OrgChartFO.xsl?>
```

Durch Aktivieren des Kontrollkästchens "Pfad relativ machen zu..." können Sie den Pfad der zugewiesenen Datei relativ machen. Ein XML-Dokument kann zwei zugewiesene XSLT-Dateien haben: Eine Standard XSLT-Transformationsdatei, sowie eine zweite Datei für eine XSLT-Transformation in FO.

13.7.7 XML-Beispieldatei zuweisen

Mit diesem Befehl weisen Sie einem XSLT-Dokument eine **XML-Datei** zu. Der Befehl fügt eine Verarbeitungsanweisung ein, die eine XML-Datei spezifiziert, die mit dieser XSLT-Datei verarbeitet werden soll, wenn die XSL-Transformation an der XSLT-Datei ausgeführt wird:

```
<?altova_samplexml C:\workarea\html2xml\article.xml?>
```

Beachten Sie bitte, dass Sie den Pfad der zugewiesenen Datei relativ machen können. Aktivieren Sie dazu das Kontrollkästchen "Pfad relativ machen zu...".

13.7.8 Gehe zu XSL

Mit diesem Befehl öffnen Sie das zugehörige XSLT Dokument, falls Ihr XML-Dokument eine Processing Instruction zur Verarbeitung eines Stylesheet (also eine XSLT-Zuweisung) wie z.B. die folgende enthält:

```
<?xml-stylesheet type="text/xsl" href="Company.xsl"?>
```

Der Befehl **Gehe zu XSL** öffnet das XSLT-Dokument anschließend in XMLSpy 2005.

13.7.9 Debugger starten

Alt+F11

© 2005 Altova GmbH

Mit diesem Befehl starten Sie den Debugger oder fahren mit der Verarbeitung des XSLT/XQuery-Dokuments fort. Wenn Breakpoints gesetzt wurden, wird der Ablauf an dieser Stelle angehalten. Wenn die Debugger-Sitzung noch nicht gestartet wurde, wird diese mit diesem Befehl gestartet und am ersten Node angehalten, um diesen zu verarbeiten. Wenn die Sitzung läuft, wird das XSLT/XQuery-Dokument bis zum Ende verarbeitet oder bis das Programm auf den nächsten Breakpoint stößt.

13.7.10 Debugger anhalten

Mit diesem Befehl stoppen Sie den Debugger. Dies ist nicht dasselbe, wie das Stoppen der Debugger-Sitzung, in der der Debugger gerade läuft. Dieser Befehl eignet sich, wenn Sie ein Dokument während einer Debugger-Sitzung bearbeiten oder andere Dateien in derselben Debugger-Sitzung verwenden möchten. Wenn Sie den Debugger angehalten haben, müssen Sie ihn erneut starten, um am Anfang des XSLT/XQuery-Dokuments zu beginnen.

13.7.11 Debugger neu starten

Mit diesem Befehl wird der XSLT Debugger mit den aktuell ausgewählten Dateien erneut gestartet und das Ausgabefenster wird gelöscht.

13.7.12 Debugger-Sitzung beenden

Mit diesem Befehl beenden Sie die Debugger-Sitzung und kehren zur normalen XMLSpy 2005 Ansicht zurück, die vor dem Start des Debuggers aktiv war. Ob die für die Debugger-Sitzung geöffneten Ausgabedokumente geöffnet bleiben oder nicht, hängt davon ab, welche Einstellung Sie im Dialogfeld XSLT/XQuery Debugger-Einstellungen gewählt haben.

13.7.13 Einsteigen

F11

Mit diesem Befehl gehen Sie Schritt für Schritt durch alle Nodes und XPath-Ausdrücke des Stylesheets. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem dieser angehalten wurde.

13.7.14 Aussteigen

Umschalt+F11

Mit diesem Befehl steigen Sie aus dem aktuellen Node aus und gehen zum nächsten **Zwilling** des Parent-Node oder zum nächsten Node auf der nächsthöheren Ebene über der des Parent-Node.

13.7.15 Überspringen

Strg+F11

Mit diesem Befehl überspringen Sie den aktuellen Node und gehen zum nächsten Node auf derselben Ebene oder zur nächsthöheren Ebene über der des aktuellen Node. Dieser Befehl dient auch zum erneuten Starten des Debuggers, nachdem er angehalten wurde.

13.7.16 Aktuellen Execution Node einblenden

Dieser Befehl dient zum Anzeigen/Auswählen des aktuellen Execution Node im XSLT/XQuery-Dokument und des entsprechenden Context Node im XML-Dokument. Dies ist nützlich, wenn Sie auf andere Register geklickt haben, auf denen bestimmter Code im XSLT Stylesheet bzw. in der XML-Datei angezeigt oder markiert wird, und anschließend wieder zum Ausgangspunkt zurückkehren möchten.

13.7.17 Breakpoint einfügen/löschen

F9

Dieser Befehl dient zum Einfügen oder Löschen eines Breakpoint an der angeklickten Position. Sie können sowohl für Nodes im XSLT/XQuery-Dokument als auch im XML-Dokument Breakpoints innerhalb einer Zeile definieren, um festzulegen, wo die Verarbeitung unterbrochen werden soll. An der Stelle, an der Sie den Breakpoint setzen, erscheint oberhalb des Nodes eine strichlierte rote Linie. An schließenden Nodes können keine Breakpoints definiert werden, und Breakpoints an Attributen in XSLT-Dokumenten werden ignoriert. Dieser Befehl steht auch zur Verfügung, wenn Sie mit der rechten Maustaste auf die gewünschte Stelle für den Breakpoint klicken.

13.7.18 Breakpoint ein/aus

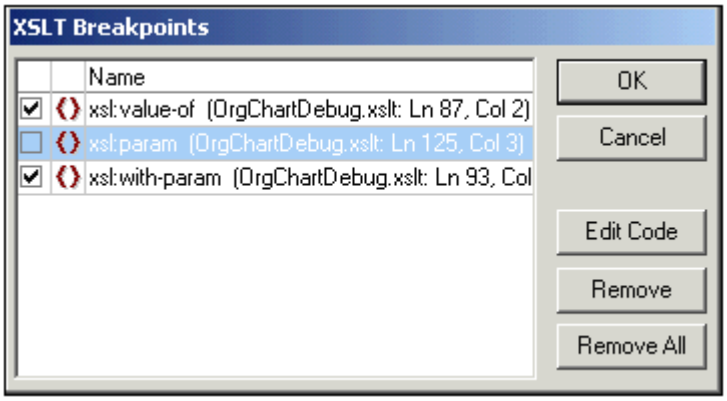
Strg+F9

Mit diesem Befehl (es gibt keine Symbolleisten-Schaltfläche) schalten Sie bereits definierte Breakpoints ein oder aus. Eingeschaltete Breakpoints werden in rot, ausgeschaltete Breakpoints werden in grau angezeigt. Der Debugger hält bei ausgeschalteten Breakpoints nicht an. Um einen Breakpoint ein- bzw. auszuschalten, setzen Sie den Cursor in den Node-Namen und klicken Sie auf "Breakpoint ein/aus". Dieser Befehl kann auch über das Kontextmenü (rechte Maustaste) aufgerufen werden.

13.7.19 Breakpoints...

Mit diesem Befehl öffnen Sie das Dialogfeld "XSLT/XQuery Breakpoints", in welchem eine Liste zuvor definierter Breakpoints (einschl. aller deaktivierten Breaktpoints) in allen Dateien in der aktuellen Debugger-Sitzung angezeigt wird.

© 2005 Altova GmbH



Das Kontrollkästchen zeigt an, ob ein Breaktpoint aktiv (Häkchen) oder deaktiviert ist. Um einen markierten Breakpoint zu entfernen, klicken Sie auf die Schaltfläche **Entfernen.** Zum Entfernen aller Breakpoints klicken Sie auf **Alle löschen.** Wenn Sie auf die Schaltfläche **Code bearbeiten** klicken, wird der entsprechende Breakpoint in der Datei direkt aufgerufen.

13.7.20 Debug-Fenster

Wenn Sie den Cursor über den Befehl **Debug-Fenster** platzieren, erscheint ein Untermenü mit einer Liste der verschiedenen Informationsfenster des XSLT/XQuery-Debuggers. Wenn Sie eines der Informationsfenster aus dieser Liste auswählen, wird dieses Fenster auf der XSLT/XQuery Debugger-Oberfläche markiert. Dieser Befehl kann nur während einer laufenden Debugger-Sitzung verwendet werden.

13.7.21 XSLT/XQuery-Einstellungen

Mit diesem Befehl rufen Sie das Dialogfeld "XSLT/XQuery-Einstellungen" auf, in welchem Sie Benutzereinstellungen für den Debugger festlegen können. Nähere Informationen dazu finden Sie im Abschnitt XSLT/XQuery Debugger.

13.8 Menü "Authentic"

Über die Authentic-Ansicht können Sie XML-Dokumente editieren, **die auf StyleVision Power Stylesheets basieren, die mit StyleVision von Altova erstellt worden sind!** Diese Stylesheets enthalten alle Informationen, die benötigt werden, damit eine XML-Datei grafisch in der Authentic-Ansicht dargestellt werden kann. Zusätzlich zu diesen Anzeigeinformationen ermöglichen Ihnen StyleVision Power Stylesheets außerdem, Daten in eine XML-Datei zu schreiben. Diese Daten werden mit Hilfe aller XSLT-Stylesheet-Funktionen dynamisch verarbeitet und sofort in Form einer Authentic-Ansicht ausgegeben.

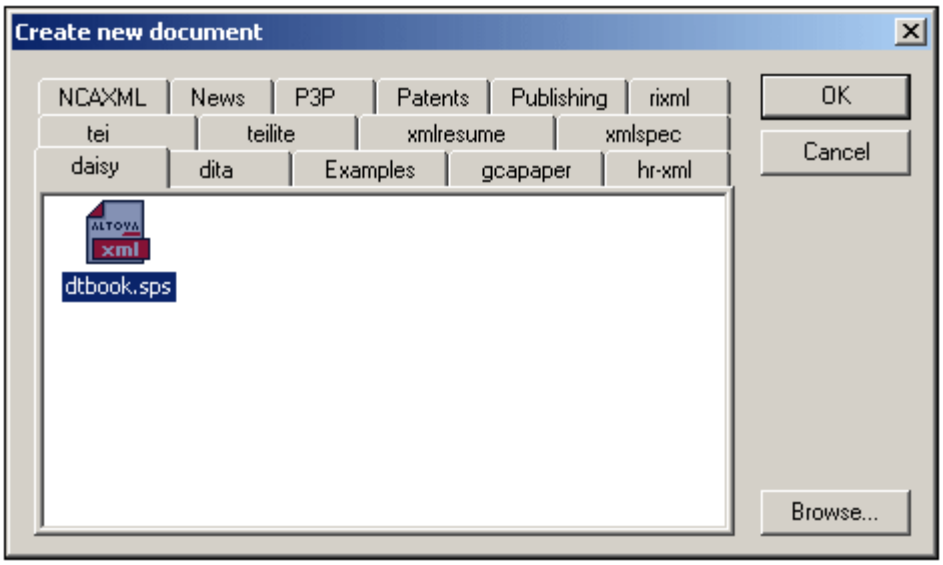
Des Weiteren können Sie StyleVision Power Stylesheets erstellen, um eine editierbare XML-Ansicht einer Datenbank anzuzeigen. Das StyleVision Power Stylesheet enthält Informationen, um eine Verbindung zu einer Datenbank herzustellen, die Datenbankdaten in der Authentic-Ansicht anzuzeigen und wieder zurück in die Datenbank zu schreiben.

Das Menü "Authentic" enthält Befehle zum Editieren von XML-Dokumenten in der Authentic-Ansicht. Ein Tutorial zur Authentic-Ansicht finden Sie im Abschnitt Tutorials.

13.8.1 Neues Dokument...

Mit dem Befehl **Neues Dokument...** können Sie eine neue XML-Dokumentenvorlage in der Authentic-Ansicht öffnen. Die XML-Dokumentenvorlage basiert auf einem StyleVision Power Stylesheet (`.sps`-Datei) und wird durch Auswahl des StyleVision Power Stylesheet geöffnet.

Wenn Sie den Befehl **Neues Dokument...** auswählen, wird das Dialogfeld "Neues Dokument anlegen" angezeigt.



Suchen Sie die gewünschte SPS-Datei und wählen Sie diese aus. Daraufhin wird eine XML-Dokumentenvorlage in der Authentic-Ansicht geöffnet.

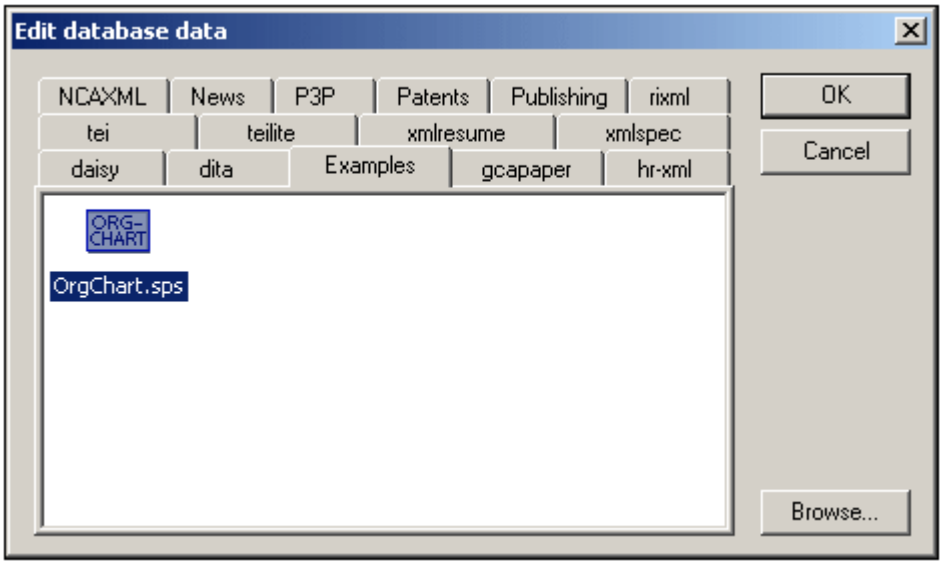
Hinweis: StyleVision Power Stylesheets werden mit Hilfe von Altova StyleVision 2005 erstellt. Dem StyleVision Power Stylesheet ist eine XML-Vorlagendatei zugewiesen. Die Daten in dieser XML-Datei liefern die Ausgangsdaten für die neue Dokumentenvorlage, die in der Authentic-Ansicht geöffnet wird.

© 2005 Altova GmbH

13.8.2 Datenbankdaten bearbeiten

Mit dem Befehl **Datenbankdaten bearbeiten...** öffnen Sie eine editierbare Ansicht einer Datenbank (DB) in der Authentic-Ansicht. Alle Informationen über die Herstellung einer Verbindung zur Datenbank, die Anzeige der DB und die Übernahme von Änderungen an der Datenbank über die Authentic-Ansicht befinden sich in einem StyleVision Power Stylesheet. Mit dem Befehl **Datenbankdaten bearbeiten...** öffnen Sie ein solches DB-basiertes StyleVision Power Stylesheet. Damit stellen Sie eine Verbindung zur DB her und zeigen die DB-Daten (durch eine XML-Linse) in der Authentic-Ansicht an.

Bei Aufruf des Befehls **Datenbankdaten bearbeiten...** erscheint das Dialogfeld "Datenbankdaten bearbeiten".



Suchen Sie die gewünschte SPS-Datei und wählen Sie diese aus. Diese Datei stellt die Verbindung zur DB her und öffnet eine editierbare Ansicht der Daten in der Authentic-Ansicht.

Warnung: Wenn Sie versuchen, ein StyleVision Power Stylesheet mit dem Befehl **Datenbankdaten bearbeiten...** zu öffnen, das nicht auf einer DB basiert, oder wenn Sie versuchen, ein DB-basiertes StyleVision Power Stylesheet zu öffnen, das in einer StyleVision-Version vor der Version StyleVision 2005 erstellt wurde, erhalten Sie eine Fehlermeldung.

Hinweis: StyleVision Power Stylesheets werden mit Altova StyleVision 2005 erstellt.

13.8.3 StyleVision Stylesheet zuweisen

Mit diesem Befehl weisen Sie einem **XML-Dokument** ein StyleVision Power Stylesheet (SPS) zu, um die Ansicht und Bearbeitung in der Authentic-Ansicht zu ermöglichen. Das StyleVision Power Stylesheet, das der XML-Datei zugewiesen werden soll, muss auf demselben Schema basieren, auf dem auch die XML-Datei basiert.

Um einer XML-Datei ein StyleVision Power Stylesheet zuzuweisen, machen Sie die XML-Datei zur aktiven und wählen Sie den Befehl **Stylevision Stylesheet zuweisen...**. Daraufhin wird ein Dialogfeld geöffnet, in dem Sie angeben können, welche StyleVision Power Stylesheet-Datei zugewiesen werden soll. Wählen Sie die gewünschte Datei aus und klicken Sie auf OK, um die gewünschte SPS-Anweisung in Ihr XML-Dokument einzufügen. Beachten Sie bitte, dass Sie den Pfad der zugewiesenen Datei relativ machen können, indem Sie das Kontrollkästchen "Pfad

relativ machen zu ..." aktivieren.

```
<?xml version="1.0" encoding="UTF-8"?>
<?altova_sps HTML-Orgchart.sps?>
```

Im obigen Beispiel ist der Name des StyleVision Power Stylesheet `HTML_Orgchart.sps`, und die Datei befindet sich im selben Verzeichnis wie die XML-Datei.

Hinweis: In früheren Altova-Produktversionen wurde eine Verarbeitungsanweisung mit dem Ziel bzw. Namen `xmlspysps` verwendet. Eine Verarbeitungsanweisung sah daher etwa folgendermaßen aus: `<?xmlspysps HTML-Orgchart.sps?>`. Diese älteren Verarbeitungsanweisungen sind auch in aktuellen Altova-Produktversionen bei Authentic immer noch gültig.

13.8.4 StyleVision Stylesheet bearbeiten

Mit diesem Befehl wird StyleVision gestartet, so dass das StyleVision Power Stylesheet sofort editiert werden kann.

13.8.5 XML Entities definieren

Unabhängig davon, ob Ihr Dokument auf einer DTD oder einem XML-Schema basiert, können Sie für die Authentic-Ansicht Entitäten definieren, die dann in der Eingabehilfe "Entities" und im Kontext-Untermenü **Entity einfügen** angezeigt werden. Wenn Sie in der Eingabehilfe "Entities" auf eine Entität doppelklicken, wird die Entität an der aktuellen Cursorposition eingefügt.

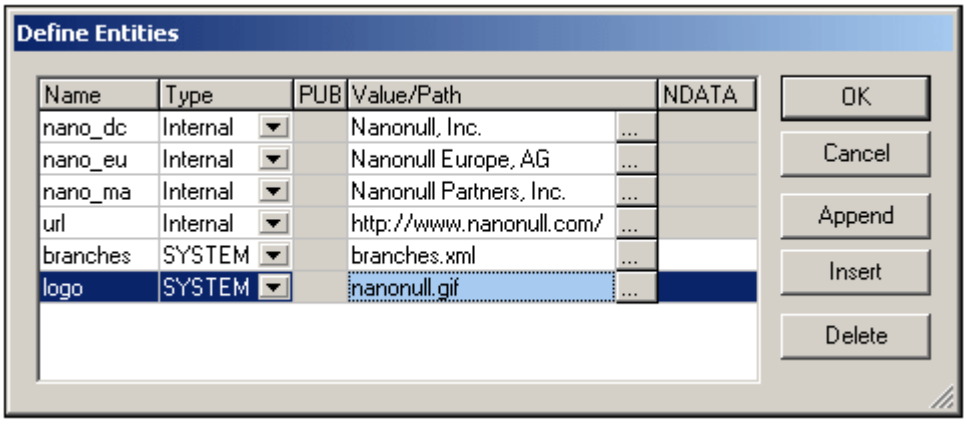
Eine Entität ist nützlich, wenn Sie einen Textstring, ein XML-Fragment oder eine andere externe Ressource an mehreren Stellen in Ihrem Dokument einfügen möchten. Sie definieren die Entität, die im Grund eine Kurzbezeichnung ist, die die gewünschten Daten repräsentiert, im Dialogfeld "Entities definieren". Nachdem Sie die Entität definiert haben, können Sie sie an mehreren Stellen in Ihrem Dokument einfügen. Dies spart Zeit und ist effizient, wenn Sie Änderungen am Dokument vornehmen möchten.

Sie können in Ihrem Dokument zwei Arten von Entitäten verwenden: **geparste** Entitäten, wobei es sich um XML-Daten handelt (entweder Textstrings oder ein Fragment eines XML-Dokuments) oder **nicht geparste** Entitäten, wobei es sich um nicht-XML-Daten wie z.B. eine Binärdatei handelt (gewöhnlich Grafiken, Sound oder Multimedia-Objekte). Jede Entität hat einen Namen und einen Wert. Bei geparsten Entitäten ist die Entität ein Platzhalter für die XML-Daten. Der Wert der Entität sind entweder die XML-Daten selbst oder eine URI, die auf eine XML-Datei verweist, die die XML-Daten enthält. Bei nicht geparsten Entitäten ist der Wert der Entität eine URI, die auf die nicht-XML-Datendatei verweist.

Um eine Entität zu definieren, gehen Sie vor wie folgt:

1. Wählen Sie **Authentic | XML Entities definieren...**. Dadurch wird das Dialogfeld "Entities definieren" geöffnet.

© 2005 Altova GmbH



2. Geben Sie in das Feld "Name" den Namen der Entität ein. Dies ist der Name, der in der Eingabehilfe "Entities" angezeigt werden wird.
3. Wählen Sie im Feld "Type" den Entitätstyp aus der Dropdown-Liste aus. Es stehen drei Typen zur Auswahl. Bei einer **internen** Entität handelt es sich um Text, der im XML-Dokument selbst gespeichert ist. Bei Auswahl von **PUBLIC** oder **SYSTEM** geben Sie an, dass sich die Ressource außerhalb der XML-Datei befindet und mit Hilfe eines Public Identifier bzw. eines System Identifier aufgerufen wird. Ein System Identifier ist eine URI, die den Speicherort einer Ressource angibt. Der Public Identifier ist ein pfadunabhängiger Identifier, der es einigen Prozessoren erlaubt, die Ressource zu identifizieren. Wenn Sie sowohl einen Public Identifier als auch einen System Identifier definieren, wird der Public Identifier mit dem System Identifier aufgelöst und der System Identifier wird verwendet.
4. Wenn Sie als Typ PUBLIC ausgewählt haben, geben Sie den Public Identifier Ihrer Ressource im Feld PUBLIC ein. Wenn Sie Internal oder SYSTEM als Typ ausgewählt haben, wird das Feld PUBLIC deaktiviert.
5. Im Feld "Value/Path" haben Sie folgende Eingabemöglichkeiten:

   - Ist der Entitätstyp "Internal", geben Sie den gewünschten Textstring als Wert der Entität ein. Verwenden Sie zur Begrenzung der Entität keine Anführungszeichen, da diese als Teil des Textstrings behandelt werden. Beachten Sie, dass Entitäten sich gut zum Einfügen von Unicode-Zeichen in Ihr Dokument eignen; geben Sie dazu die Unicode-Nummer als Wert der internen Entität ein.
   - Ist der Entitätstyp SYSTEM, geben Sie die URI der Ressource an oder wählen Sie mit Hilfe der Schaltfläche zum Durchsuchen eine Ressource in Ihrem lokalen Netzwerk aus. Wenn die Ressource geparste Daten enthält, muss es eine XML-Datei sein (d.h. sie muss eine XML-Erweiterung haben). Andernfalls kann als Ressource eine Binärdatei wie z.B. eine GIF-Datei verwendet werden.
   - Ist der Entitätstyp PUBLIC, müssen Sie in diesem Feld zusätzlich noch einen System Identifier eingeben.

6. Der Eintrag NDATA sagt dem Prozessor, dass diese Entität nicht geparst, sondern an den entsprechenden Prozessor gesendet werden soll. Das Feld NDATA sollte daher nur bei nicht geparsten Entitäten verwendet werden.

Dialogfeldfunktionen

Sie können Entitäten anhängen, einfügen und löschen, indem Sie auf die entsprechende Schaltfläche klicken. Des Weiteren können Sie Entitäten spaltenweise alphabetisch sortieren, indem Sie auf die Kopfzeile der jeweiligen Spalte klicken; bei einmaligem Klicken werden die Entitäten aufsteigend sortiert, bei zweimaligem Klicken absteigend. Die Größe des Dialogfelds und die Spaltenbreite lassen sich anpassen.

Einschränkungen

- Eine in einer anderen Entität enthaltene Entität wird weder im Dialogfeld, noch der

Authentic-Ansicht noch der XSLT-Ausgabe aufgelöst. Das kaufmännische Und-Zeichen einer solchen Entität wird in seiner Escape-Form also als `&amp angezeigt.`

- Externe Entitäten werden in der Authentic-Ansicht nicht aufgelöst, außer es handelt sich bei der Entität um eine Bilddatei, die als Wert eines Attributs eingegeben wurde, dessen Typ im Schema als `ENTITY` oder `ENTITIES` definiert wurde. Solche Entitäten werden aufgelöst, wenn das Dokument mit einem in der Authentic Stylesheet generierten XSLT verarbeitet wird.

13.8.6 Markupsymbole ausblenden

Mit diesem Befehl werden die Markup-Symbole in der Authentic-Ansicht ausgeblendet.

13.8.7 Kleine Markup-Symbole einblenden

Mit diesem Befehl werden kleine Markup-Symbole in der Authentic-Ansicht angezeigt.

13.8.8 Große Markup-Symbole einblenden

Mit diesem Befehl werden große Markup Symbole in der Authentic-Ansicht angezeigt.

13.8.9 Gemischte Markup-Symbole einblenden

Mit diesem Befehl werden gemischte Markup Symbole in der Authentic-Ansicht angezeigt.

13.8.10 Zeile anhängen

Mit diesem Befehl wird eine Zeile an die Tabelle in der Authentic-Ansicht angehängt.

13.8.11 Zeile einfügen

Mit diesem Befehl wird eine Zeile in die Tabelle in der Authentic-Ansicht eingefügt.

13.8.12 Zeile kopieren

Mit diesem Befehl wird die markierte Zeile in der Tabelle in der Authentic-Ansicht dupliziert.

© 2005 Altova GmbH

13.8.13 Zeile nach oben

Mit diesem Befehl wird in der Authentic-Ansicht die markierte Zeile um eine Zeile nach oben verschoben.

13.8.14 Zeile nach unten

Mit diesem Befehl wird in der Authentic-Ansicht die markierte Zeile um eine Zeile nach unten verschoben.

13.8.15 Zeile löschen

Mit diesem Befehl wird in der Authentic-Ansicht die markierte Zeile aus der Tabelle gelöscht.

13.9 Menü "Konvertieren"

Mit XMLSpy 2005 stehen Ihnen leistungsfähige Datenaustauschfunktionen zur Verfügung:

- Importieren und Exportieren von Text-, Word-Prozessor-, Datenbanken- und XML-Dateien.
- Erzeugen von XML-Schemas, die auf einer bereits existierenden Datenbank **basieren**.
- Importieren von Datenbankdaten, die auf einem existierenden XML Schema basieren.
- Erzeugen einer **Datenbankstruktur**, die auf einem existierenden Schema basiert.

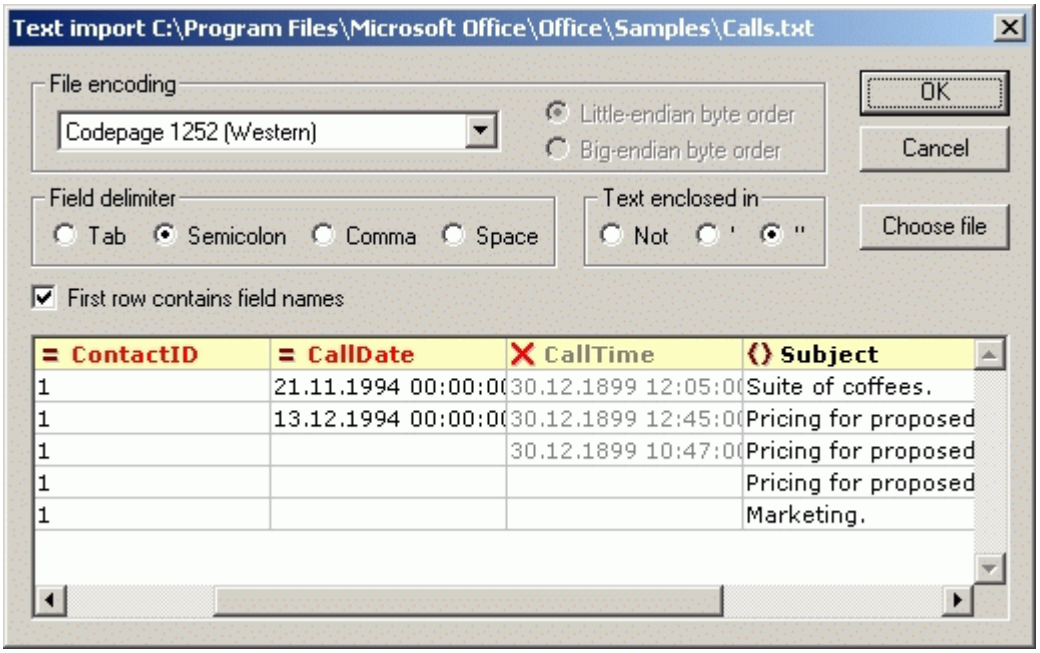
XMLSpy 2005 unterstützt derzeit die folgenden Datenbanken und Verbindungsmethoden:

- Microsoft Access (ADO)
- Microsoft SQL Server (ADO)
- Oracle (OCI)
- MySQL (ODBC)
- Sybase (ODBC)
- IBM DB2 (ODBC)

- Jede ADO-Datenbank
- Jede ODBC-Datenbank

13.9.1 Textdatei importieren...

Mit diesem Befehl **importieren** Sie beliebige Dateien mit **strukturiertem Text** in XMLSpy 2005 und konvertieren diese sofort in ein XML Format. Dies ist besonders beim Importieren von bereits vorhandenen Daten aus älteren Systemen hilfreich, da die meisten Softwareprodukte eine Schnittstelle zum Exportieren von Text beinhalten.



Unicode Konvertierung
Um Daten in Unicode (Grundlage aller XML-Dokumente) zu konvertieren, spezifizieren Sie bitte,

in welchem Zeichensatz die Datei derzeit kodiert ist.

Westeuropäische und US-amerikanische Windows-Systeme arbeiten meist mit Codepage 1252, auch unter dem Namen ANSI bekannt. Beim Importieren von 16-bit oder 32-bit Unicode (UCS-2, UTF-16, oder UCS-4) Dateien, können Sie weiters zwischen Little-endian und Big-endian Bytefolge wählen.

Feldbegrenzung
Um Textdateien erfolgreich zu importieren, müssen Sie angeben, wie die Feldbegrenzung auszusehen hat, die verwendet werden soll, um Spalten und Felder innerhalb der Datei zu begrenzen. XMLSpy 2005 erkennt automatisch die gebräuchlichen Feldbegrenzungen (CR, LF, oder CR+LF).

Textbegrenzung
Textdateien, die aus älteren Systemen exportiert wurden, enthalten oftmals Textdaten, die zwischen Anführungszeichen (") stehen, um diese besser von numerischen Werten abzuheben. Hier können Sie angeben, welche Anführungszeichen verwendet werden sollen, und ob diese beim Exportieren gelöscht werden sollen.

Erste Zeile enthält Feldnamen
Bei Textdateien werden oftmals in der ersten Zeile die Feldnamen der Datei angezeigt. Mit XMLSpy 2005 können Sie den XML Element- oder Attributnamen nach diesen Informationen definieren.

Datei wählen
Klicken Sie auf die Schaltfläche "Datei wählen" um nach der Angabe der Importparameter eine spezifische Datei zu importieren. Der Dateiname wird gespeichert, und beim nächsten Aufrufen dieser Funktion im Dialogfeld als Auswahl angezeigt. So können Sie die Einstellungen der Datei ändern und diese neu importieren, falls die resultierende XML-Datei nicht Ihren Vorstellungen entspricht.

Nachdem Sie die zu importierende Datei ausgewählt haben, erhalten Sie eine Vorschau der zu importierenden Daten. Wenn Sie jetzt Änderungen der oben genannten Optionen vornehmen, werden diese sofort in die Vorschau übernommen.

Namen der Felder oder Spalten ändern
Durch Klicken auf den Feld- oder Spaltennamen können Sie diesen ändern.

Attribute, Elemente, oder ohne Spalten
Mit XMLSpy 2005 haben Sie die Wahl zwischen dem Import einer Spalte als Element oder Attribut. Weiters können Sie auswählen, ob die Spalte übersprungen werden soll.

Klicken Sie auf das links neben dem Spaltentitel stehende Symbol, um zwischen den Optionen hin- und herzuschalten. Im oben genannten Beispiel, werden
- ContactID und CallDate - als Attribute importiert
- CallTime - nicht importiert
- Subject - als Element importiert.

13.9.2 Datenbank importieren...

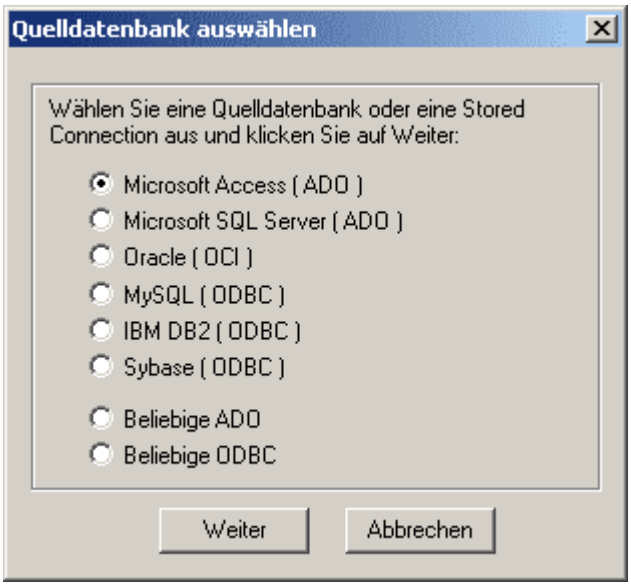
Mit diesem Befehl können Sie **Datenbankdaten** aus verschiedenen Quellen importieren. Es kann jedoch immer nur **eine Tabelle** auf einmal importiert werden.

Die im Folgenden beschriebene Methode ist für alle der anschließend beschriebenen Optionen des Menüs "Konvertieren" und für alle Arten von Datenbankverbindungen praktisch dieselbe:

- Importieren Sie Datenbankdaten
- Erstellen Sie anhand der DB-Struktur ein XML-Schema
- DB-Import auf Basis des XML-Schemas
- Erstellen der DB-Struktur anhand des XML-Schemas

Wenn wir die Menüoption "**Datenbank importieren...**" als Beispiel nehmen, so ist der Ablauf der folgende:

1. Wählen Sie die Menüoption **Konvertieren | Datenbank importieren**.
   Daraufhin wird das Dialogfeld "Quelldatenbank auswählen" geöffnet, in dem Sie die Datenbankquelle auswählen.



2. Klicken Sie auf die Datenbank, die Sie als Datenquelle verwenden möchten (z.B. MS Access) und anschließend auf die Schaltfläche "Weiter", um die Verbindung zur ausgewählten Datenbank herzustellen.
3. Klicken Sie auf einen der unten stehenden Links, um zu sehen, wie Sie mit der jeweiligen Datenbank fortfahren:

   MS Access (ADO)
   MS SQL Server (ADO)
   Oracle (OCI)
   IBM DB2 (ODBC)

4. Klicken Sie auf die Schaltfläche "Import", um die Datenbankdaten zu importieren.

Bitte beachten Sie:
Der Ablauf ist beim Herstellen der Verbindung bei Verwendung von ODBC bei allen Datenbanken (MySQL, Sybase und IBM DB2) immer derselbe.

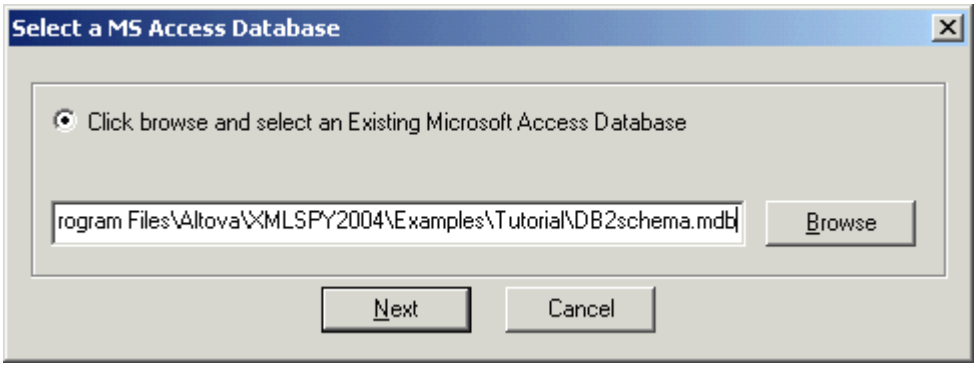
MS Access als Quelle

XMLSpy 2005 unterstützt die Datenbanken MS Access 2000 und MS Access 2003.

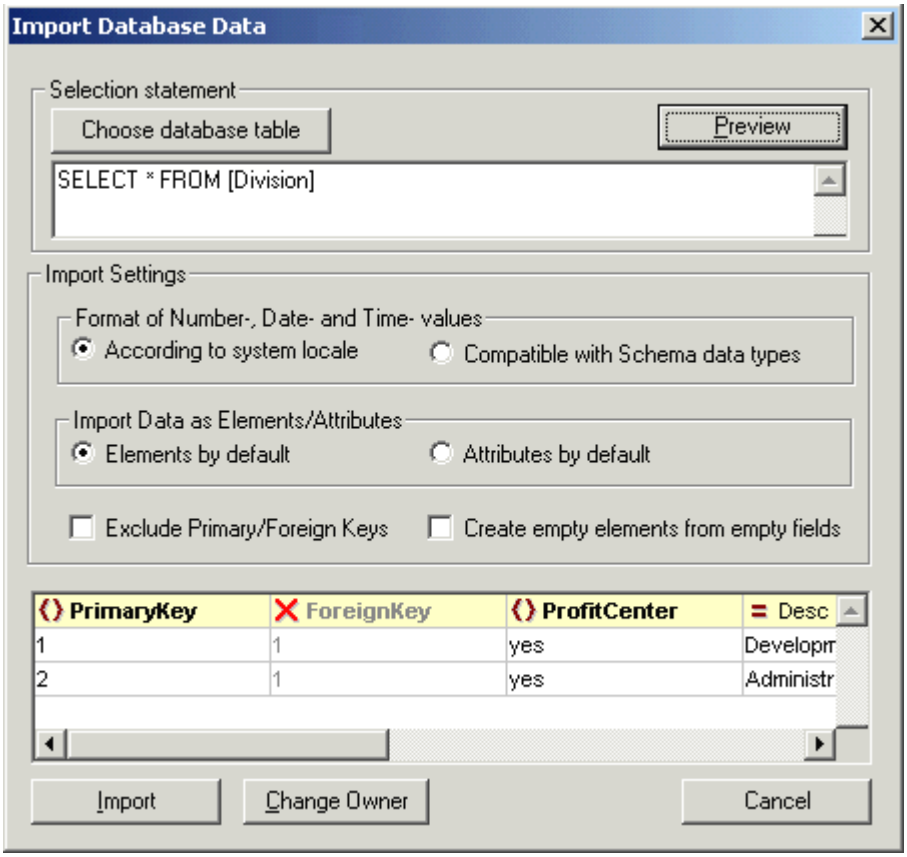
1. Nachdem Sie "**Microsoft Access**" ausgewählt haben und auf "Weiter" geklickt haben,
2. Klicken Sie auf die Schaltfläche **Durchsuchen**, um die entsprechende Datenbank auszuwählen, und anschließend auf Öffnen, um sie zu öffnen und in dieses Dialogfeld zurückzukehren.

© 2005 Altova GmbH

Der Name und Pfad der Datenbank werden nun im Textfeld angezeigt.



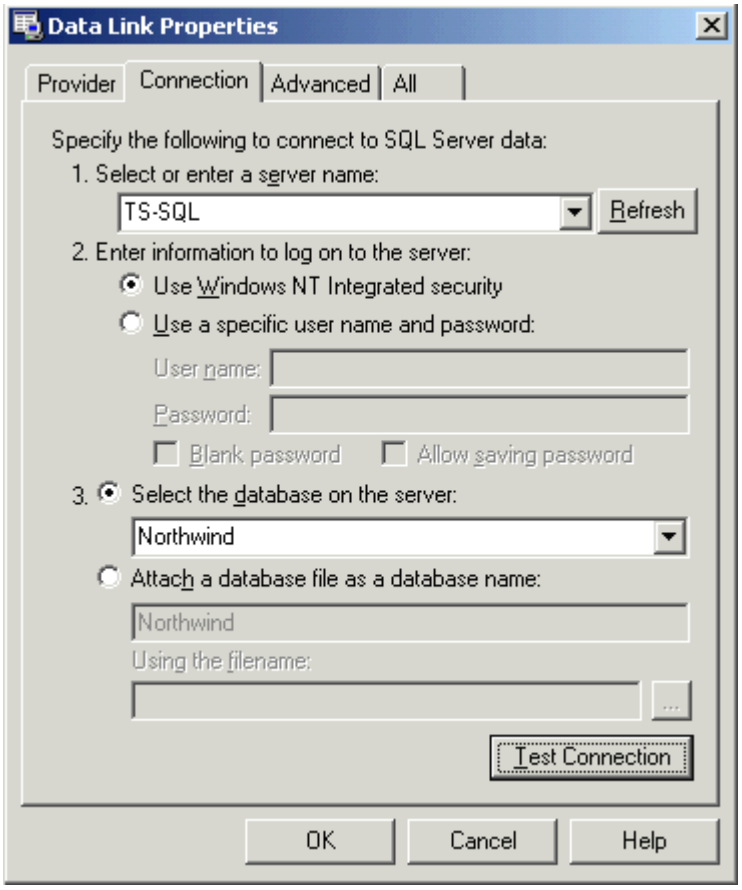
3. Klicken Sie zum Fortfahren auf "Weiter".
   Das Dialogfeld "Datenbank importieren" wird geöffnet.



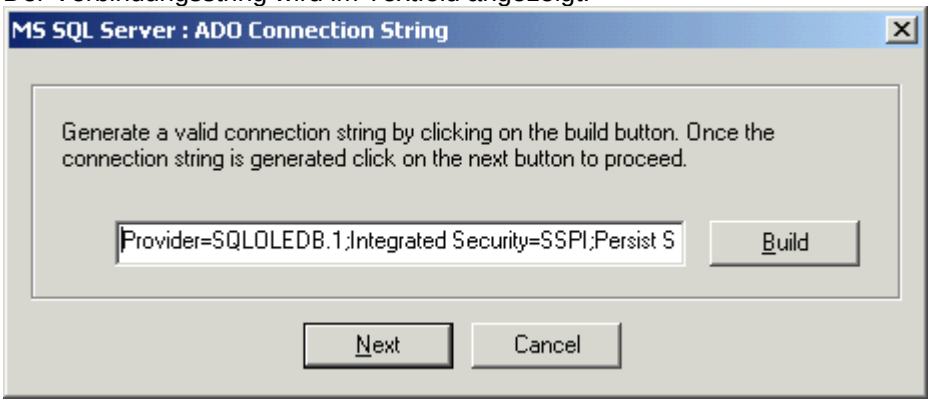
4. Klicken Sie auf die Schaltfläche **Datenbank-Tabelle wählen** und wählen Sie die zu importierende Tabelle aus (z.B. Division).
   Die Select-Anweisung wird im Textfeld angezeigt.
5. Klicken Sie auf die Schaltfläche **Vorschau,** um eine Vorschau der Datenbankfelder anzuzeigen.
   Sie können nun die Importeinstellungen näher definieren, indem Sie in der Gruppe Importeinstellungen die jeweiligen Optionen auswählen.
6. Klicken Sie auf die Schaltfläche **Import**, um die Datenbank zu importieren.

MS SQL als Quelle

1. Nach Auswahl von "**Microsoft SQL Server**" und Klicken auf "Weiter" wird das Dialogfeld "MS SQL Server : ADO connection String" geöffnet.
2. Klicken Sie auf die Schaltfläche **Erzeugen**, um mit der Definition des ADO Connection String zu beginnen.
   Das Dialogfeld "**Data Link Properties**" wird geöffnet, in dem Sie den Verbindungsserver, die Login-Eigenschaften usw. definieren können.
3. Füllen Sie die benötigten Felder aus und klicken Sie auf die Schaltfläche "Test connection".



4. Klicken Sie auf OK, um die Definition des Verbindungsstrings abzuschließen. Der Verbindungsstring wird im Textfeld angezeigt.

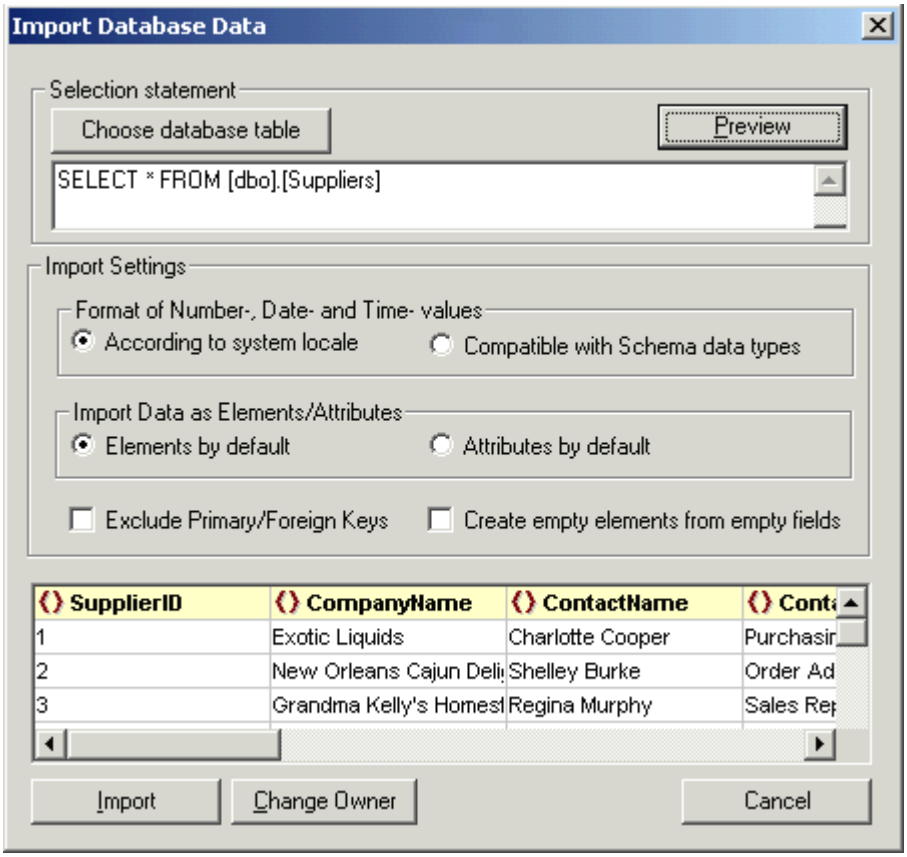


5. Klicken Sie zum Fortfahren auf "Weiter".

© 2005 Altova GmbH

Daraufhin wird das Dialogfeld "**Datenbank importieren**" geöffnet.

6. Klicken Sie auf die Schaltfläche **Datenbank-Tabelle wählen** und wählen Sie die zu importierende Tabelle aus (z.B. suppliers).
   Die Select-Anweisung wird im Textfeld angezeigt.
7. Klicken Sie auf die Schaltfläche **Vorschau,** um eine Vorschau der Datenbankfelder anzuzeigen.



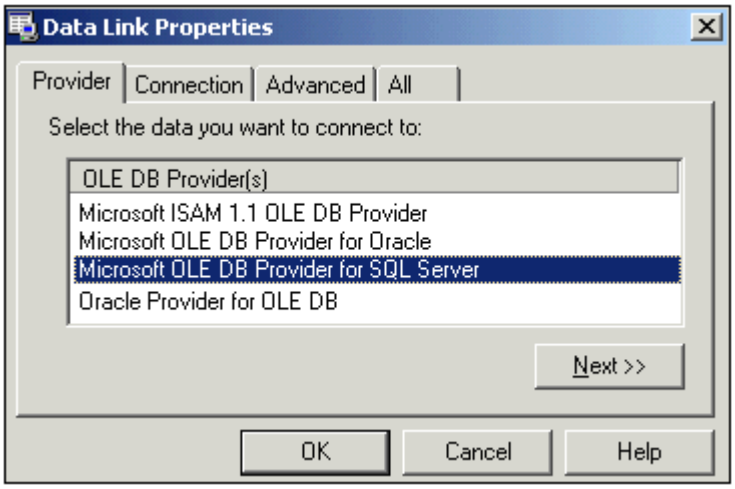
Sie können nun die Importeinstellungen näher definieren, indem Sie in der Gruppe Importeinstellungen die jeweiligen Optionen auswählen.

8. Klicken Sie auf die Schaltfläche **Import**, um die Datenbank zu importieren.

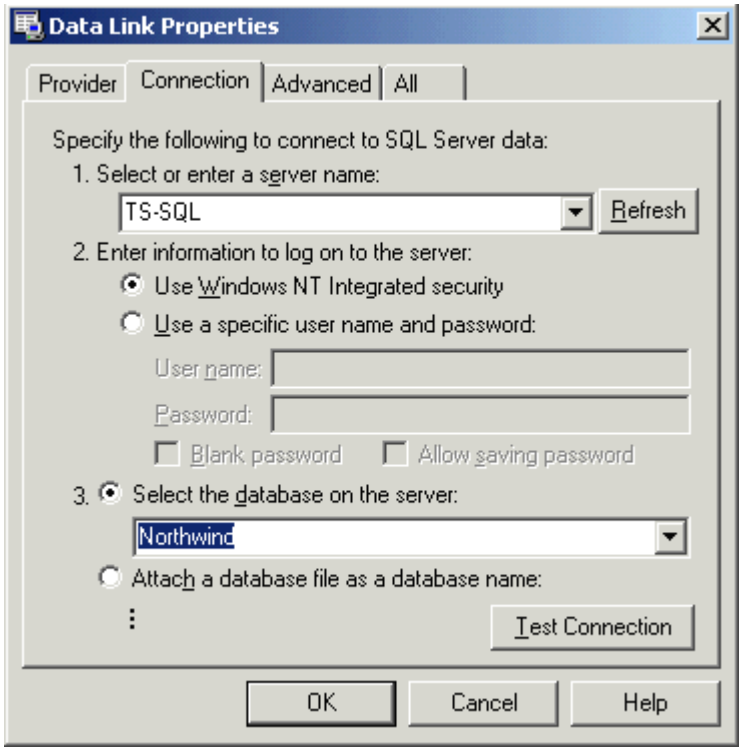
ADO Connection String

Wenn Sie für die Eingabe des ADO Connection Strings Hilfe benötigen, klicken Sie auf die Schaltfläche "Erzeugen". Das Dialogfeld "Datenverknüpfungseigenschaften" wird geöffnet.

Dialogfeld Data Link Properties:



Wählen Sie den entsprechenden OLE DB Provider aus der Liste, klicken Sie auf die Schaltfläche Weiter, und geben Sie die Verbindungseigenschaften ein:



Für die Datenbank-Verbindung wird in XMLSpy 2005 meist die ActiveX Data Object (ADO) Schnittstelle verwendet. ADO ist in Windows 2000 und Microsoft Office 2000 inkludiert. Bei der Verwendung einer älteren Version von Windows oder Office müssen Sie ADO möglicherweise neu installieren.

Bitte beachten Sie:

Aktivieren Sie das Kontrollkästchen "Allow saving password" um Ihr Passwort in den Connection String einzufügen und eine Fehlermeldung bei Zustandekommen der Verbindung zu vermeiden.

© 2005 Altova GmbH

Bitte lesen Sie unsere FAQ für weitere Informationen zum Thema ADO und Herunterladen der Microsoft Data Access Components (MDAC), womit Sie zur neuesten Version von ADO upgraden können. Weiters empfehlen wir Ihnen im Internet die Seite Microsoft Universal Data Access.

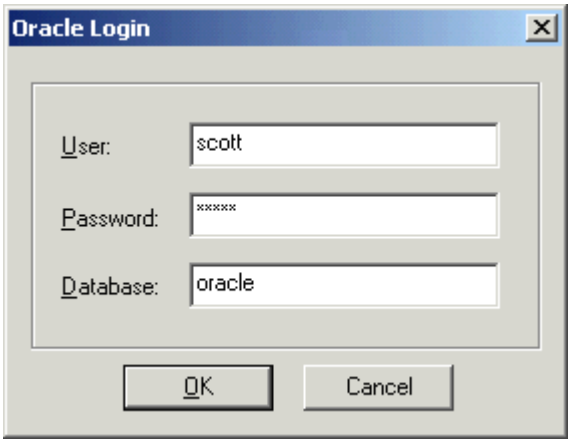
Wenn Sie Ihre Daten importiert und in XML konvertiert haben, können Sie die Funktion DTD/Schema generieren verwenden, um alle Datentyp-Informationen aus der Datenbank in Ihrem XML-Dokument zu übernehmen.

Mit XMLSpy 2005 können Sie nun eine Datenbank in ein Schema konvertieren. MS Access und weitere Datenbanken liefern automatisch die Key und Keyref Information für den ADO Treiber, der verwendet wird, um die Datenbankhierarchie zu erzeugen.

Weitere Informationen erhalten Sie im Abschnitt **Menü Konvertieren | XML-Schema von DB-Struktur erzeugen**.

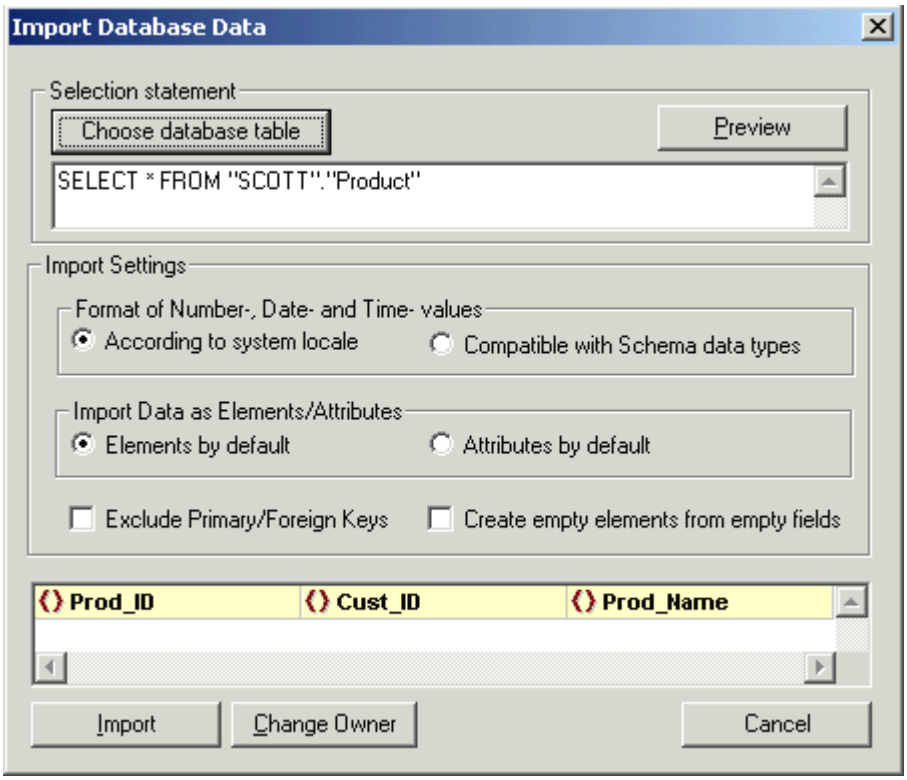
Oracle als Quelle

1. Nach Auswahl von "**Oracle (OCI)**" und Klicken auf "Weiter" wird das Dialogfeld "**Oracle Login**" geöffnet.
2. Geben Sie die Login-Parameter ein und klicken Sie auf OK.



Daraufhin wird das Dialogfeld "**Datenbank importieren**" geöffnet.

3. Klicken Sie auf die Schaltfläche **Datenbank-Tabelle wählen** und wählen Sie die zu importierende Tabelle aus.
   Die Select-Anweisung wird im Textfeld angezeigt.
4. Klicken Sie auf die Schaltfläche **Vorschau,** um eine Vorschau der Datenbankfelder anzuzeigen.



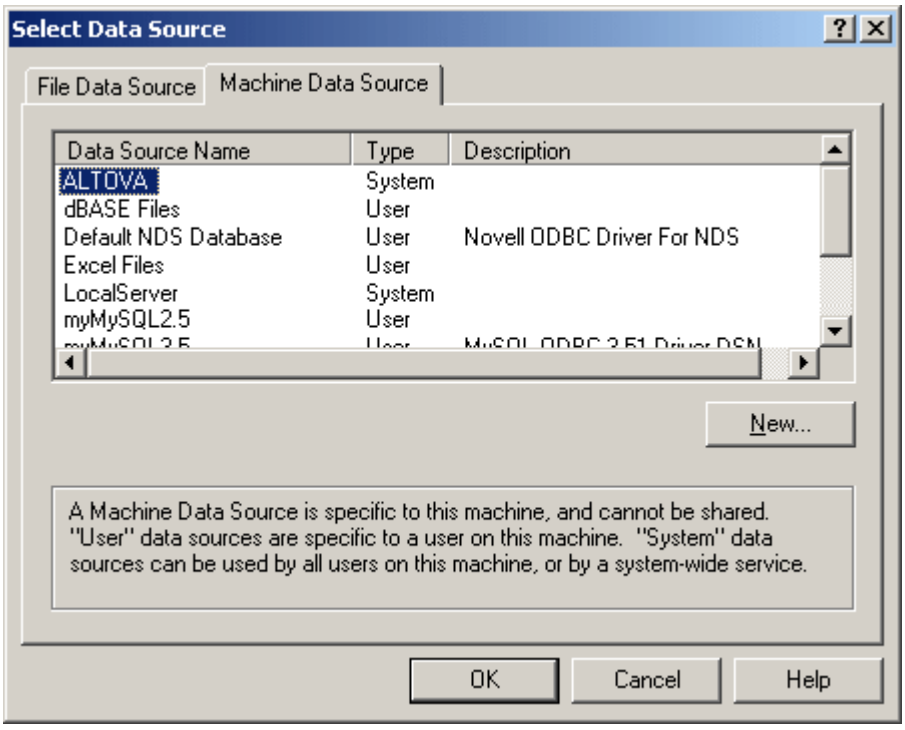
Sie können nun die Importeinstellungen näher definieren, indem Sie in der Gruppe Importeinstellungen die jeweiligen Optionen auswählen.

5. Klicken Sie auf die Schaltfläche **Import**, um die Datenbank zu importieren.

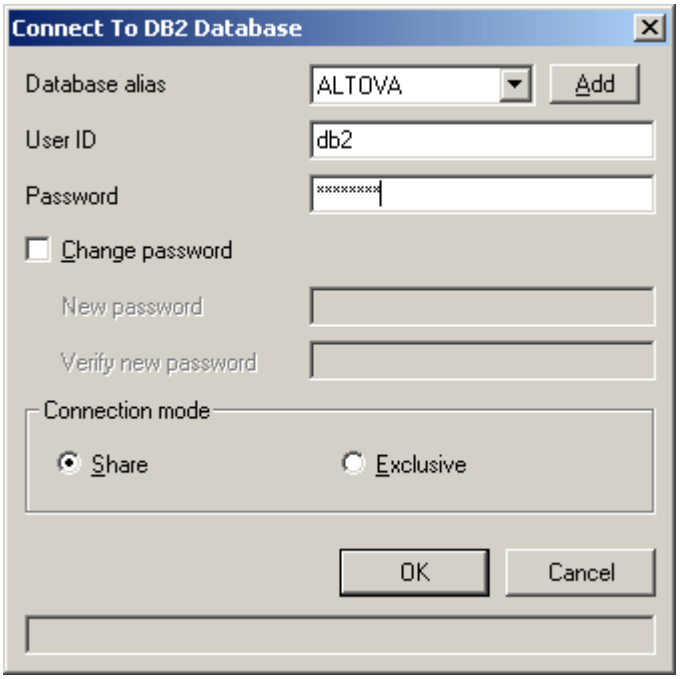
IBM DB2 als Quelle

1. Nach Auswahl von "**IBM DB2 (ODBC)**" und Klicken auf "Next" wird das Dialogfeld "**IBM DB2 : ODBC Connection String**" geöffnet.
2. Klicken Sie auf die Schaltfläche "**Erzeugen**", um mit dem Definieren des ODBC Connection String zu beginnen.
   Daraufhin wird das Dialogfeld "**Datenbank importieren**" geöffnet.

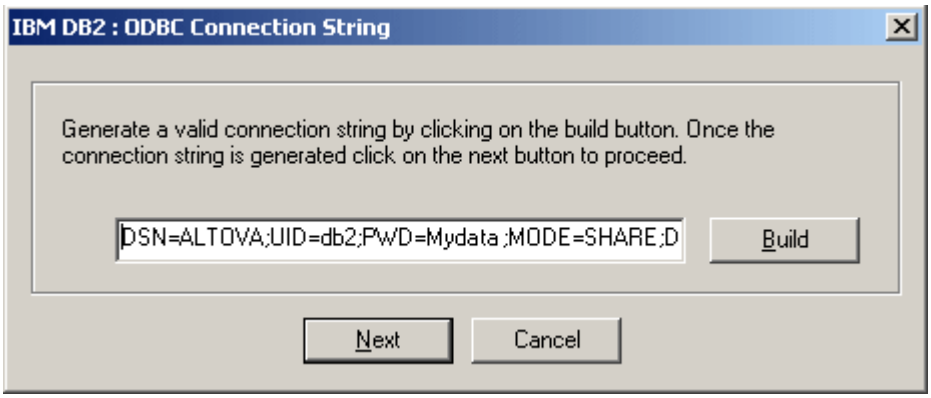
© 2005 Altova GmbH



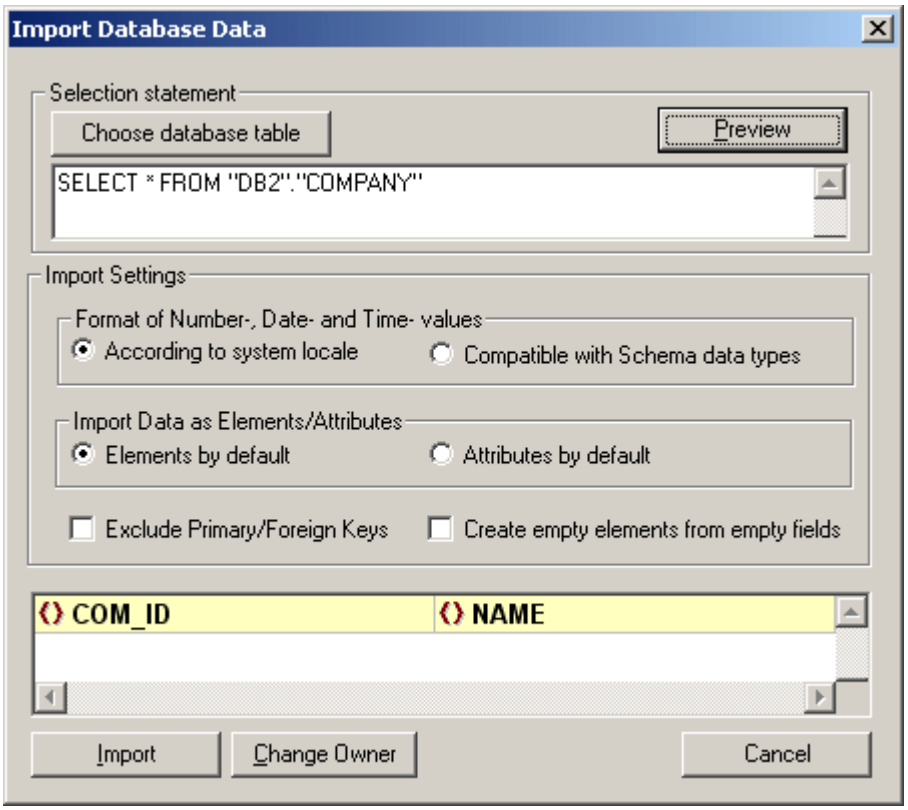
3. Klicken Sie auf das Register **Computerdatenquelle**, wählen Sie den Datenquellennamen, den Sie beim Installieren des IBM DB2 Client erstellt haben aus, und klicken Sie auf OK.
   Das Dialogfeld "Mit DB2 verbinden" wird geöffnet.
4. Geben Sie die **User ID** und das **Passwort** ein und klicken Sie auf OK, um die Verbindung zur Datenbank herzustellen.



Der Connection String wird erstellt und in das Textfeld Connection String gesetzt.



5. Klicken Sie zum Fortfahren auf "Weiter".
   Das Dialogfeld "**Datenbank importieren**" wird geöffnet.
6. Klicken Sie auf die Schaltfläche **Datenbank-Tabelle wählen** und wählen Sie die zu importierende Tabelle aus.
   Die Select-Anweisung wird im Textfeld angezeigt.
7. Klicken Sie auf die Schaltfläche **Vorschau,** um eine Vorschau der Datenbankfelder anzuzeigen.



Sie können nun die Importeinstellungen näher definieren, indem Sie in der Gruppe Importeinstellungen die jeweiligen Optionen auswählen.

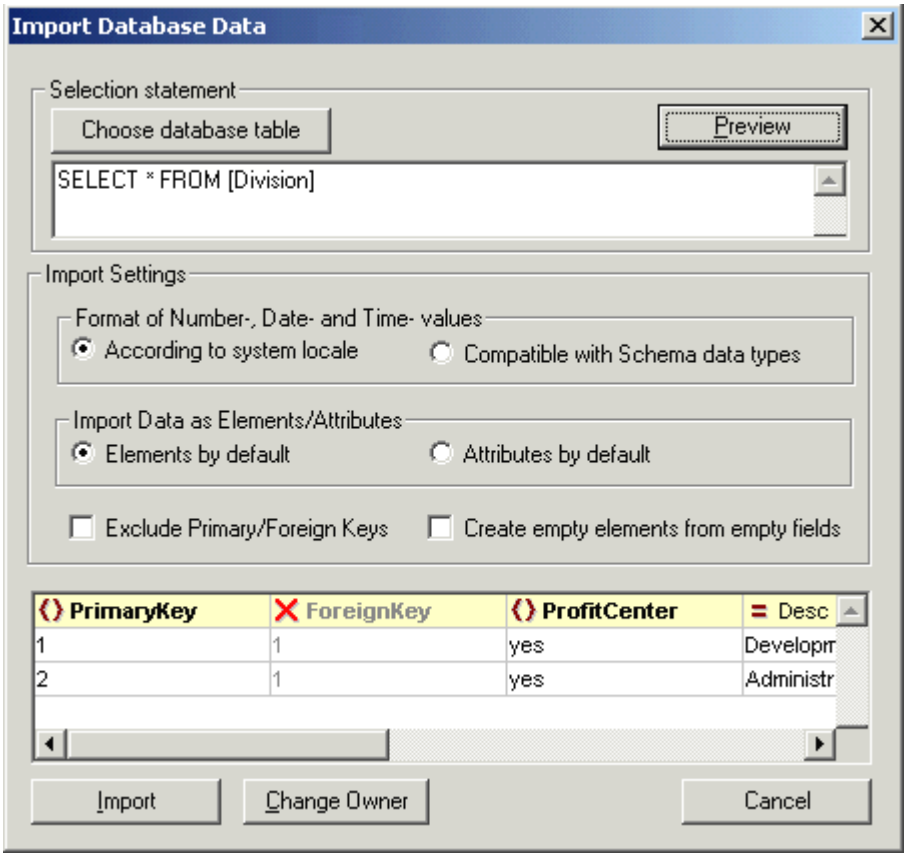
8. Klicken Sie auf die Schaltfläche **Import**, um die Datenbank zu importieren.

 © 2005 Altova GmbH

Datenbankimport-Einstellungen

Select-Anweisung

Klicken Sie zur Auswahl der Datenbank auf die Schaltfläche "Datenbank-Tabelle wählen" oder geben Sie optional dazu eine SELECT-Anweisung ein, um das zu importierende Datensatz-Set zu erstellen.



Klicken Sie nach Auswahl der Datenquelle auf die Schaltfläche **Vorschau,** um die Daten, die Sie importieren möchten, zu überprüfen.

Bitte beachten Sie:

Sie können die automatisch generierte Select-Anweisung **editieren**, wenn Sie den Importvorgang manuell steuern möchten.

Format von Zahlen-, Datums- und Zeitwerten

Mit XMLSpy 2005 können Sie verschiedene Darstellungsformen für Datums- und Zahlenformate auswählen. Sie können die resultierende XML-Datei in Verbindung mit den neuen, einheitlichen Datentypen der neuesten XML Schemavorlagen verwenden, oder die Formate beibehalten, die den lokalen Gegebenheiten angepasst sind.

Daten als Elemente/Attribute importieren

Mit diesen Optionen können Sie angeben, wie die Parent und Child Elemente importiert werden sollen: als Elemente oder Attribute. Bei der Auswahl einer dieser Optionen wird die Vorschau automatisch aktualisiert.

Primär/Sek.-Schlüssel ausschließen

Mit dieser Option können Sie die Primär- oder Sekundärschlüssel aller zu importierenden Tabellen ausschließen.

Leere Elemente aus leeren Feldern erzeugen
Mit dieser Option werden leere Elemente für jedes leere Feld der zu importierenden Tabelle erzeugt.

Attribute, Elemente, oder ohne Spalten
Mit XMLSpy 2005 haben Sie die Wahl zwischen Import einer Spalte als Element oder Attribut. Weiters können Sie auswählen, ob die Spalte übersprungen werden soll.

Klicken Sie auf das links neben dem Spaltentitel stehende Symbol, um zwischen den Optionen hin- und herzuschalten. Im oben genannten Beispiel, werden
- PrimaryKey und ProfitCenter - als Elemente importiert
- ForeignKey - nicht importiert
- Desc - als Attribut importiert.

Bitte beachten Sie:
Die hier erfolgte Auswahl wird auch auf Parents übertragen. Wenn Sie Parent-Objekte direkt ändern, so hat dies Vorrang vor den hier getätigten Einstellungen, hat allerdings nur Gültigkeit für die Parent-Elemente, und nicht für Child-Elemente.
Durch Klicken auf die Schaltfläche "Vorschau", werden die Änderungen an Parent-Elementen zurückgesetzt.

Definieren, welche Felder importiert werden sollen:
Im Fenster "Vorschau" können Sie die zu importierenden Felddaten direkt auswählen und definieren.

Durch wiederholtes **Klicken** auf das **Element-Symbol** links vom Elementnamen können Sie nacheinander die verfügbaren Auswahlmöglichkeiten aktivieren:

Definiere und importiere dieses Feld als **Element**.

Definiere und importiere dieses Feld als **Attribut**.

Überspringen, dieses Feld nicht importieren.

Change Owner
Diese Option bietet eine Liste von Ownern, die dem bei der Datenbank angemeldeten Benutzer zur Verfügung stehen. Verwenden Sie diese Option, wenn Sie einen SQL-Befehl für einen anderen Benutzer ausführen möchten.

13.9.3 Microsoft Word Dokument importieren...

Mit diesem Befehl können Sie jedes beliebige **Word Dokument importieren und in ein** XML-Format konvertieren, wenn Sie Absatzformatvorlagen in Microsoft Word verwenden. Für diese Option wird Microsoft Word oder Microsoft Office (Version 97 oder 2000) benötigt.

Mit diesem Befehl öffnen Sie das Dialogfeld "Öffnen". Wählen Sie das zu importierende Word Dokument aus.

XMLSpy 2005 erzeugt automatisch ein XML-Dokument, das CSS Stylesheet enthält. Jeder Absatz in Word generiert ein XML-Element, dessen Name der Name des entsprechenden Absatzformats in Microsoft Word ist.

© 2005 Altova GmbH

13.9.4 XML-Schema von DB-Struktur erzeugen

XMLSpy 2005 erlaubt Ihnen, ein auf einer externen Datenbankdatei basierendes Schema zu erzeugen. Es werden die folgenden Datenbanken unterstützt:

- Microsoft Access
- Microsoft SQL Server
- Oracle
- MySQL
- Sybase
- IBM DB2
- ADO-kompatible Datenbanken
- ODBC-Datenbanken

Außerdem gestattet XMLSpy 2005 das Erzeugen von zwei verschiedenen Datenbankschemas, nämlich hierarchischen und flachen.

- Im **hierarchischen** Schemamodell werden Tabellenabhängigkeiten grafisch in einer Art von Baumdiagramm-Ansicht dargestellt, in der abhängige Tabellen im Inhaltsmodell als eingerückte Child-Elemente angezeigt werden. Die Tabellenabhängigkeiten werden auch auf dem Register Identity Constraints angezeigt.
- Das **flache** Schemamodell basiert auf dem ISO-ANSI Working Draft mit dem Titel XML-Related Specification (SQL/XML) (In Zukunft als " SQL/XML Working Draft" bezeichnet). Der SQL/XML Working Draft definiert, wie Datenbanken auf XML zu mappen sind.

  Nähere Informationen zu Schemaformaten, Schema-**Erweiterungen**, Tabellen und **Identity Constraints** finden Sie unter Optionen zum Erzeugen von XML-Schemas.

Bitte beachten Sie:
Key- und Keyref-Einträge werden für die gebräuchlichsten Datenbankformate automatisch vom Befehl "DB-Struktur von XML-Schema erzeugen" erzeugt.

In der unten gezeigten Tabelle sehen Sie die erzeugten Datenbankarten, die Einschränkungen und die Verbindungsmethoden bei Verwendung der Menüoption "XML-Schema von DB-Struktur erzeugen".

	XML-Schema von DB-Struktur erzeugen				
	Direkt	mittels ODBC (eindeutige Schlüssel werden von ODBC nicht unterstützt)	mittels ADO	Unterstützung für Identity Constraint	Flaches oder hierarchisches DB-Format
Microsoft Access	OK *	OK (nicht empfohlen) Primär- und Sekundärschlüssel werden nicht unterstützt.	OK		
MS SQL Server	OK *	OK	OK		
Oracle	OK *	OK, Einschränkungen: Tabelle mit Spalten vom Typ CLOB, BLOB, BFILE; XML-Tabellen	OK, Einschränkungen: Tabellen mit Spalten vom Typ CLOB, BLOB, BFILE; XML-Tabellen; Owner-Informationen, Identity Constraints werden aus der Datenbank nicht ausgelesen		
MySQL	-	OK *	OK W		
Sybase	-	OK *	OK		
IBM DB2	-	OK *	OK		

***** **Empfohlene Verbindungsmethode für jede Datenbank.**

W **MySQL: Beim Herstellen einer ADO-Verbindung auf Basis von ODBC, wird empfohlen, entweder die Benutzer- oder die System-DSN zu verwenden.**

- **Nicht unterstützt**

Nähere Informationen dazu, wie bestimmte Datentypen auf einzelnen Schemadatentypen gemappt werden, finden Sie auch im Abschnitt "Mappen auf XML-Schema-Datentypen".

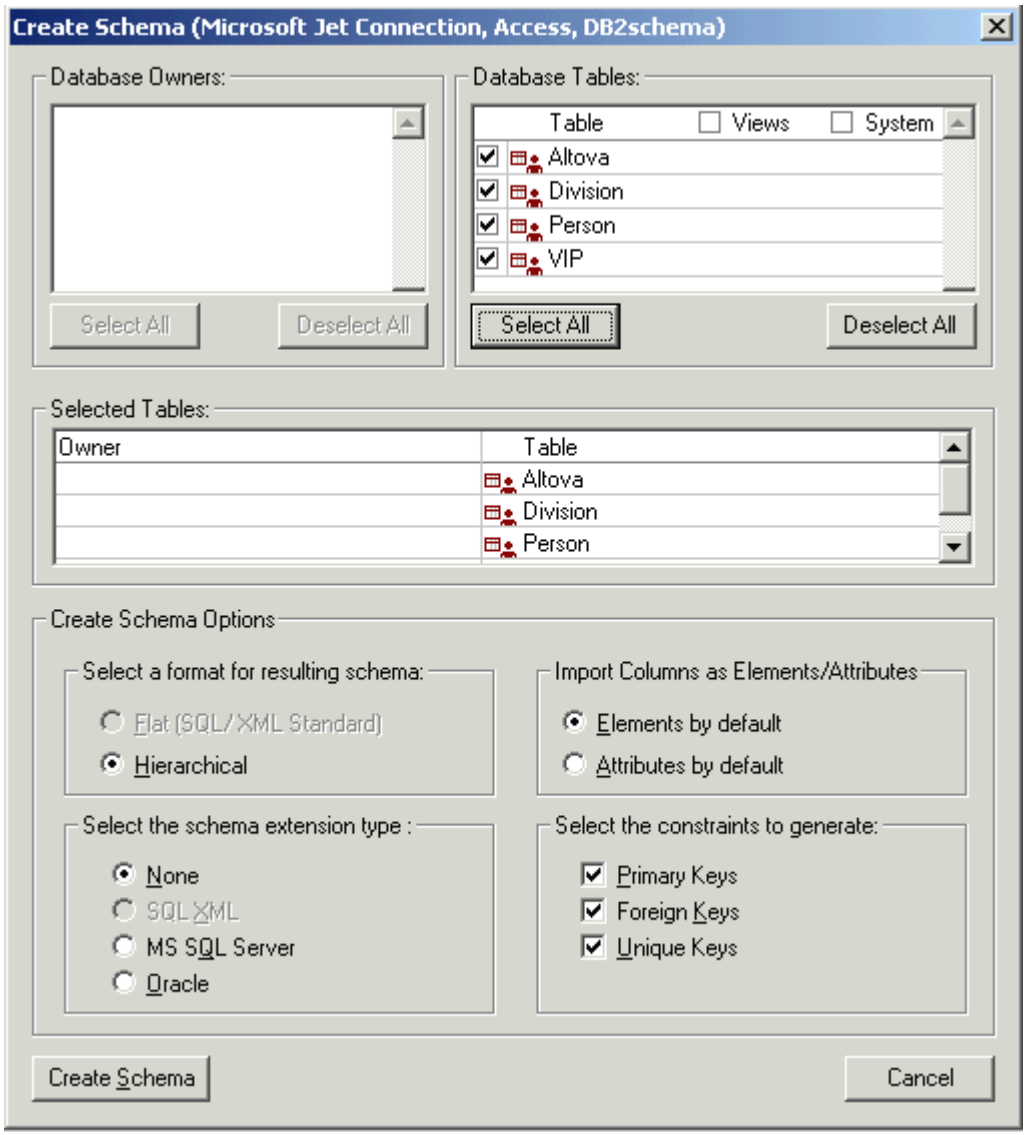
XML-Schema von DB-Struktur erzeugen (MS Access)

XMLSpy 2005 unterstützt die Datenbanken MS Access 2000 und MS Access 2003.

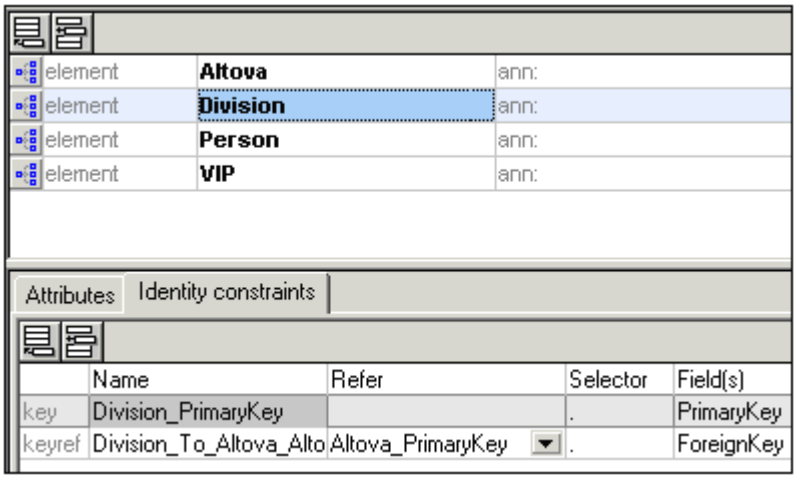
So erzeugen Sie ein hierarchisches Schema von einer Datenbankdatei (MS Access):

1. Wählen Sie die Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen.**
2. Wählen Sie **Microsoft Access** und klicken Sie auf die Schaltfläche **Weiter**.
3. Klicken Sie auf die Schaltfläche "Durchsuchen" und wählen Sie die MS Access-Quelldatenbank aus.
4. Wählen Sie die mit XMLSpy 2005 mitgelieferte Datei **DB2schema.mdb** im Ordner Tutorial aus und klicken Sie auf die Schaltfläche **Öffnen.**
5. Klicken Sie zum Fortfahren auf **Weiter.**
   Daraufhin wird das Dialogfeld **Schema erzeugen** geöffnet, in dem Sie spezifische Datenbanktabellen und andere Schemaeinstellungen definieren können.

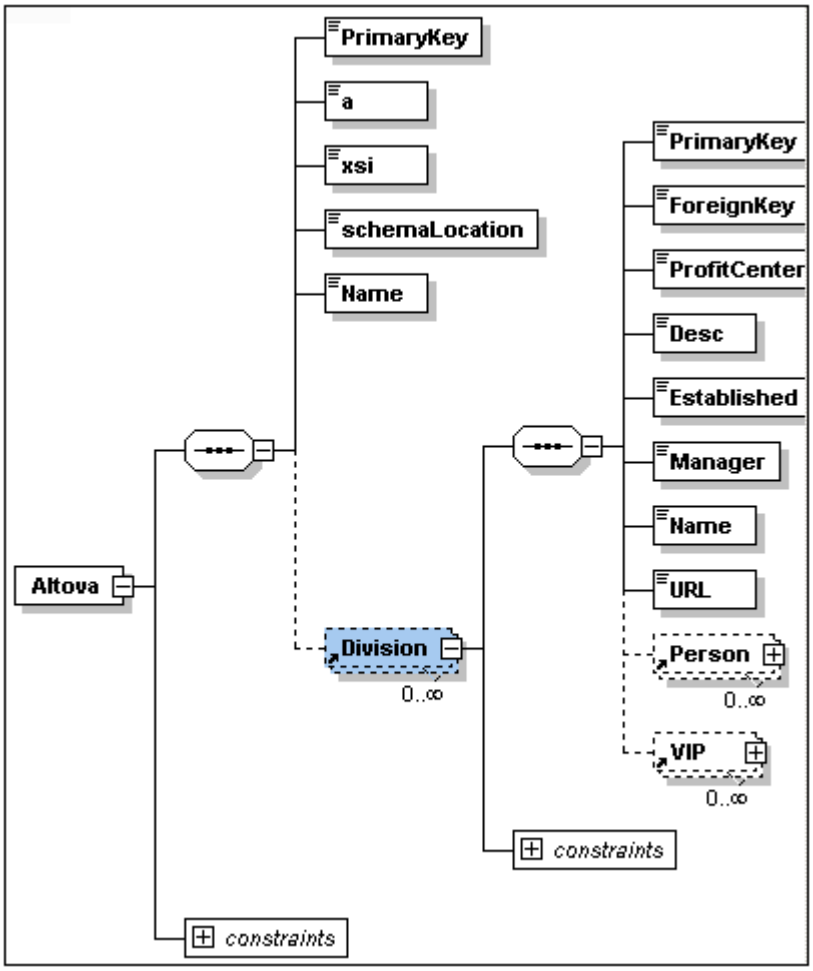
© 2005 Altova GmbH



6. Aktivieren Sie das Optionsfeld **Hierarchisch**.
7. Klicken Sie auf die Schaltfläche **Alles markieren**.
8. Klicken Sie auf die Schaltfläche **Schema erzeugen**, um die Konvertierung zu starten. Das erzeugte Schema wird in der Schema/WSDL-Design-Ansicht angezeigt. Klicken Sie auf das Register "Identity Constraints", um die keyref- und key-Felder der entsprechenden Elemente anzuzeigen.



9. Klicken Sie auf das Komponentensymbol neben dem globalen Element **Altova**, um das Inhaltsmodell dieser globalen Komponente zu sehen.



10. Wählen Sie die Menüoption **Datei | Speichern unter** und speichern Sie das neue Schema z.B. als **DB2schema.xsd**.
11. Klicken Sie auf das Symbol "Globale Elemente anzeigen", um zur Schemaübersicht zurückzukehren.

 © 2005 Altova GmbH

Bitte beachten Sie:

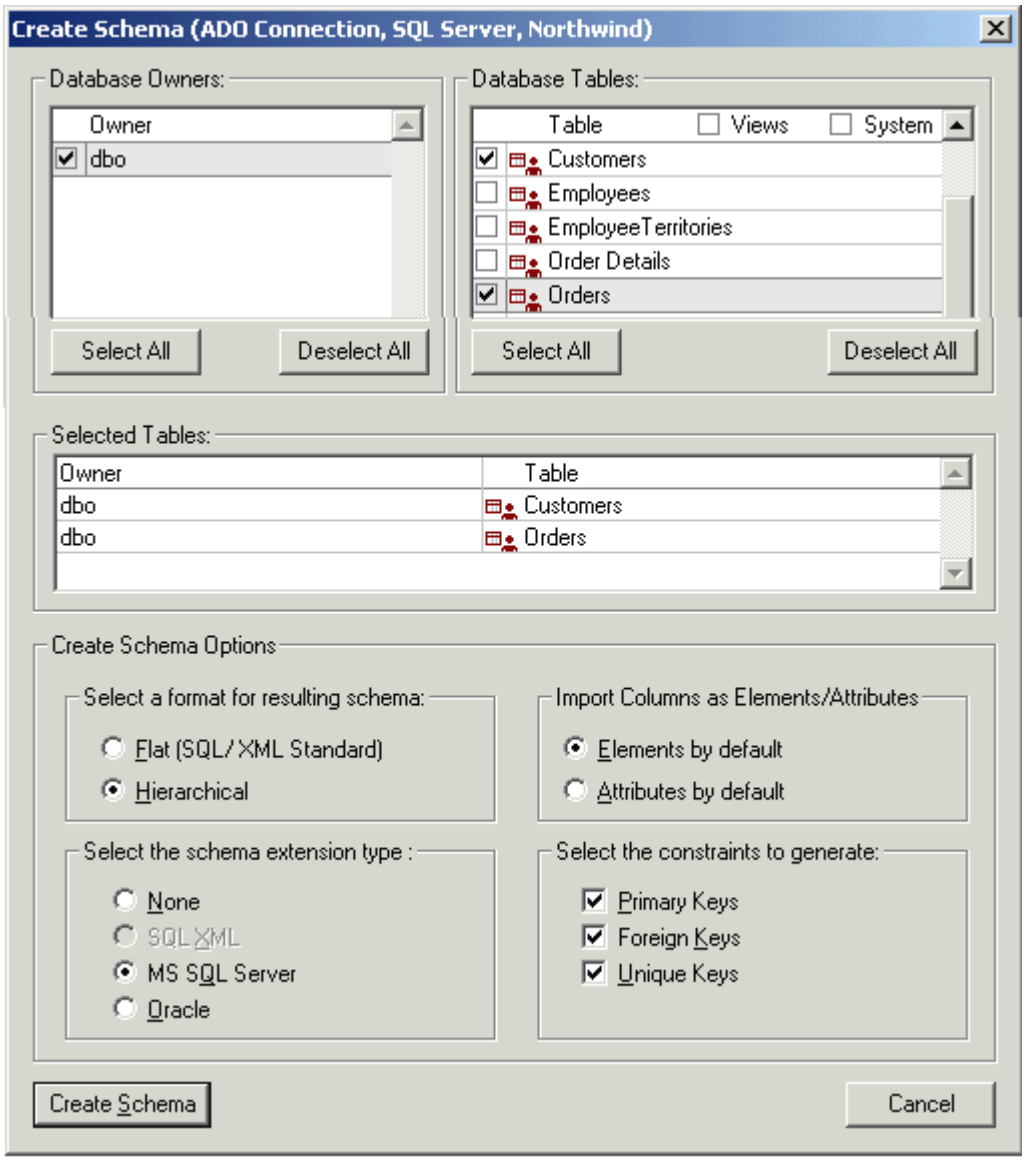
Beim Erzeugen des Schemas werden automatisch alle Namespace Präfix-Doppelpunkte in Unterstrichen-Zeichen (_) umgewandelt. Datenbanken ohne **Owner-**Informationen wie z.B. MS Access können keine Datenbanken im flachen Format erzeugen.

XML-Schema von DB-Struktur erzeugen (Microsoft SQL)

So erzeugen Sie ein hierarchisches Schema von einer Datenbankdatei (MS SQL):

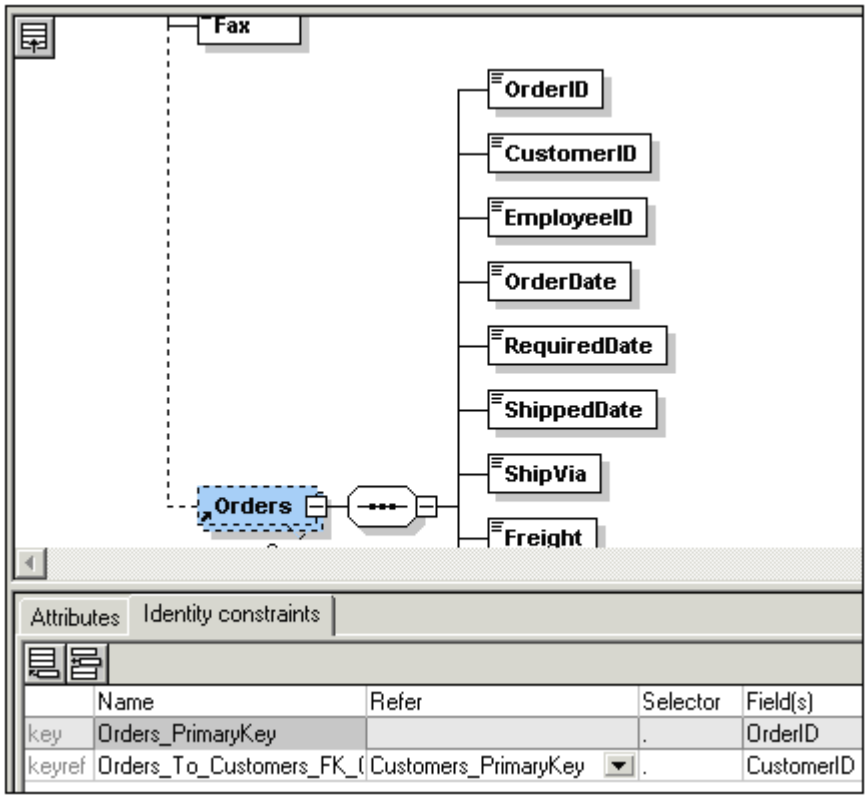
1. Wählen Sie die Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen.**
2. Wählen Sie **Microsoft SQL Server** und klicken Sie auf die Schaltfläche **Weiter**.
3. Klicken Sie auf die Schaltfläche **Erzeugen,** um die Datenbankverbindungseinstellungen auszuwählen.
   Daraufhin wird das Dialogfeld **Data Link Properties** geöffnet.
4. Klicken Sie, falls nötig, auf den Eintrag **Microsoft OLE DB Provider for SQL Server** (auf dem Register "Provider") und klicken Sie anschließend auf die Schaltfläche **Weiter**.
5. Geben Sie auf dem Register "Connection" die Daten ein, die für den Aufbau der Verbindung zur SQL-Datenbank benötigt werden, und klicken Sie anschließend auf **OK**.
   Daraufhin wird der **ADO Connection String** im Dialogfeld eingefügt.
6. Klicken Sie zum Fortfahren auf **Weiter**.
   Das Dialogfeld "Schema erzeugen" wird geöffnet.
7. Klicken Sie auf das Optionsfeld "**Hierarchisch"**.
8. Klicken Sie im Listenfeld "Datenbank-Owner" auf den **Owner** (in diesem Beispiel **dbo**) und wählen Sie die Tabellen aus, die im Schema aufscheinen sollen (in diesem Fall **Customers** und **Orders**).

   Hinweis: Die MS SQL Servererweiterungen stehen in der Eingabehilfe Details auf dem Register **SQL** zur Verfügung.

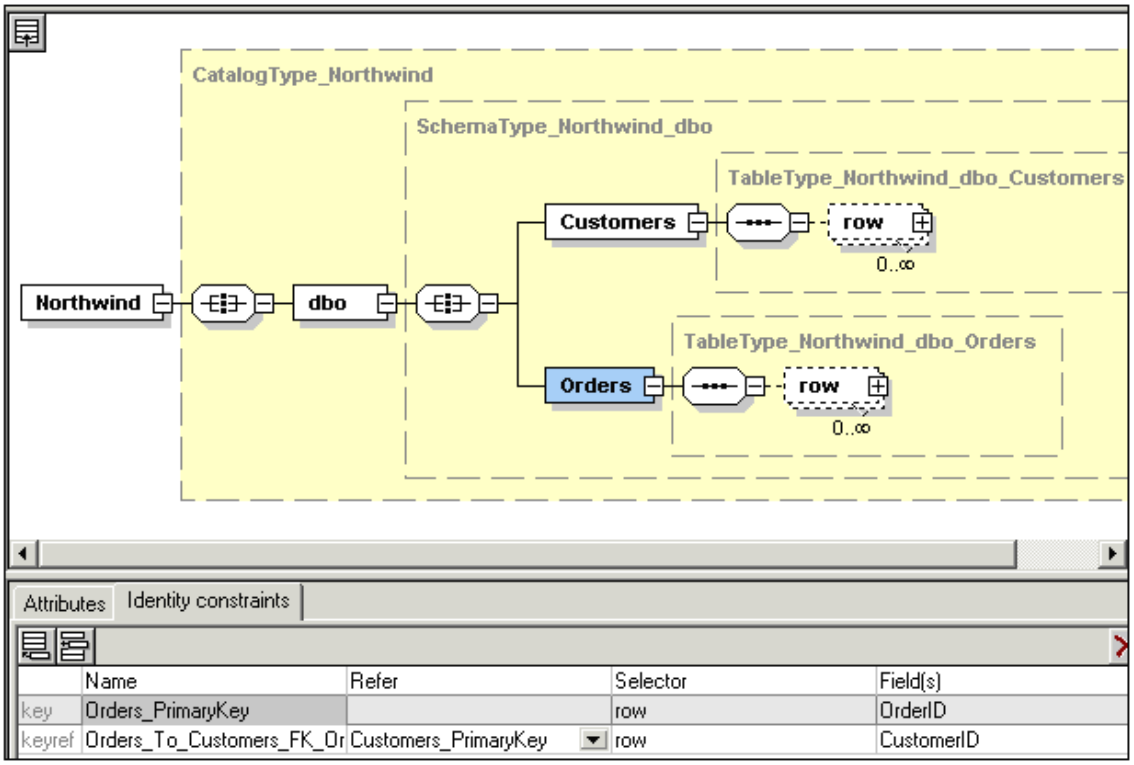


9. Klicken Sie auf die Schaltfläche **Schema erzeugen**, um die Konvertierung zu starten. Das erzeugte Schema wird in der Schema/WSDL-Design-Ansicht angezeigt.
10. Klicken Sie auf das Komponentensymbol neben dem Element **Customers**, um das Inhaltsmodell zu sehen.
    Klicken Sie auf das Register "**Identity constraints**", um die keyref- und key-Felder der entsprechenden Elemente anzuzeigen. Hinweis: Die MS SQL Servererweiterungen stehen in der Eingabehilfe Details auf dem Register **SQL** zur Verfügung.

© 2005 Altova GmbH



Ergebnis nach Generierung eines flachen Schemamodells:
Wenn Sie im Dialogfeld "Schema erzeugen" die Option "Flach (SQL/XML-Standard)" gewählt haben, sieht das erzeugte Schema wie unten gezeigt aus..

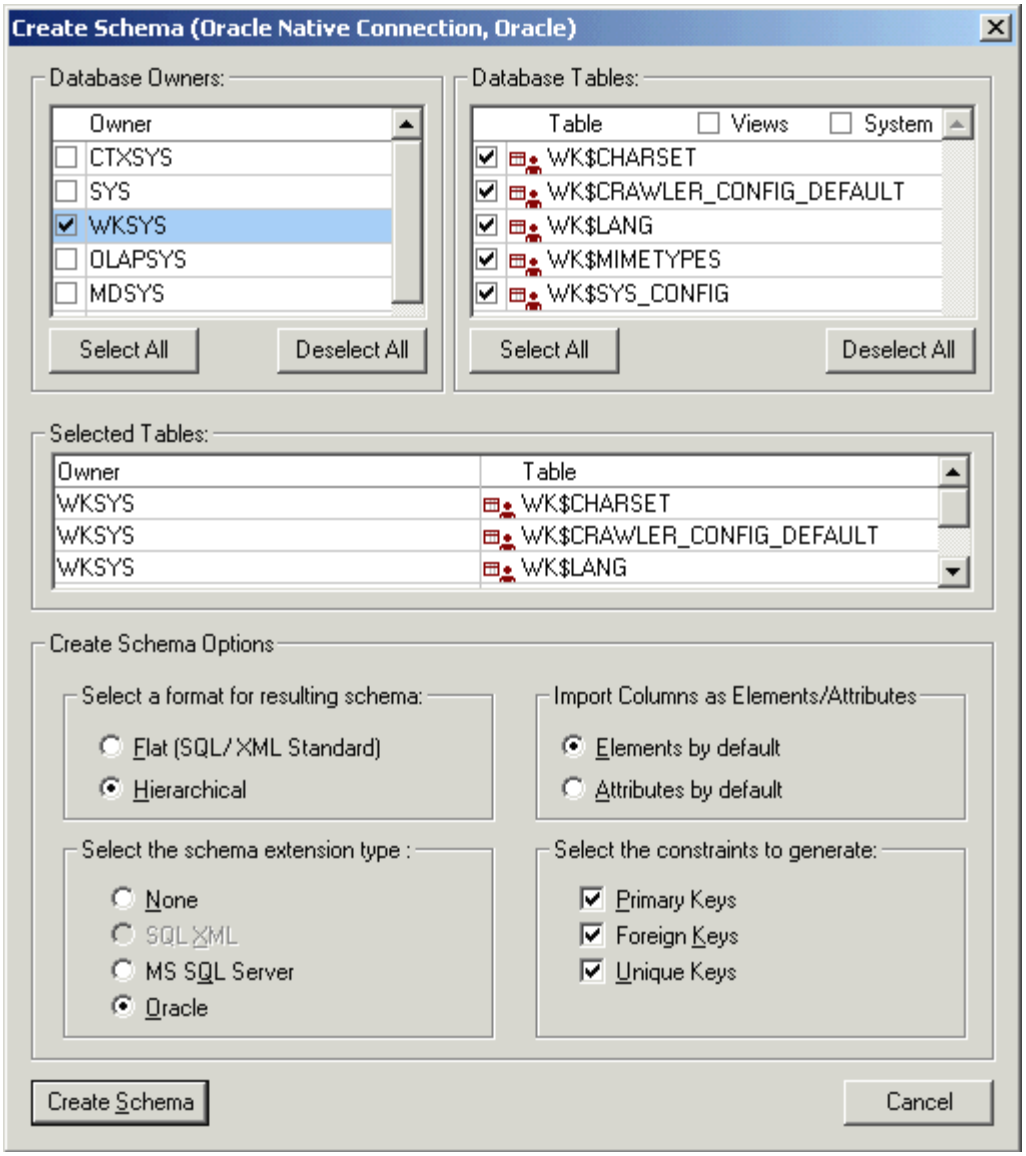


XML-Schema von DB-Struktur erzeugen (Oracle)

Um das folgende Beispiel auf Ihrem Rechner durchgehen zu können, muss auf Ihrem PC bzw. Netzwerk der Oracle Client installiert sein. Stellen Sie bitte sicher, dass Ihre Oracle Client-Installation auch '**Oracle Windows Interfaces**' enthält. Wenden Sie sich bitte an Ihren Datenbankadministrator, um den Oracle Client zu installieren.

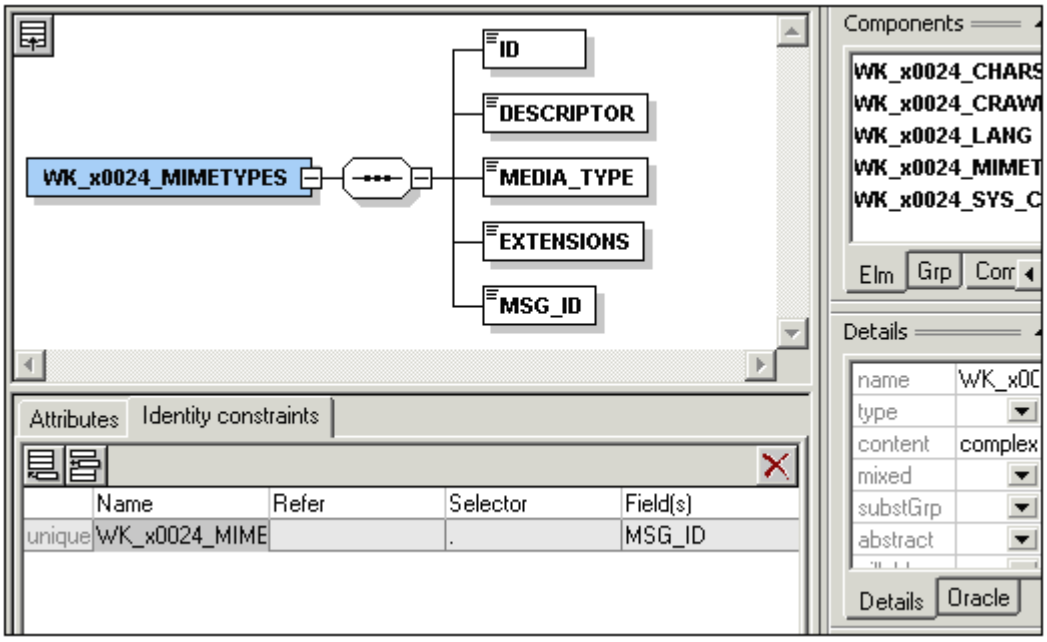
So erzeugen Sie ein hierarchisches Schema von einer Datenbankdatei (Oracle):

1. Wählen Sie die Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen.**
2. Wählen Sie **Oracle (OCI)** und klicken Sie auf die Schaltfläche **Weiter**.
   Daraufhin wird das Dialogfeld "Oracle Login" geöffnet.
3. Geben Sie die Login-Parameter ein und klicken Sie auf **OK.**



4. Klicken Sie auf das Optionsfeld "**Hierarchisch"**.
5. Klicken Sie im Listenfeld "Datenbank-Owner" auf den **Owner** (in diesem Beispiel **WKSYS**) und wählen Sie die Tabellen aus, die im Schema aufscheinen sollen.
6. Klicken Sie auf die Schaltfläche **Schema erzeugen**, um die Konvertierung zu starten.

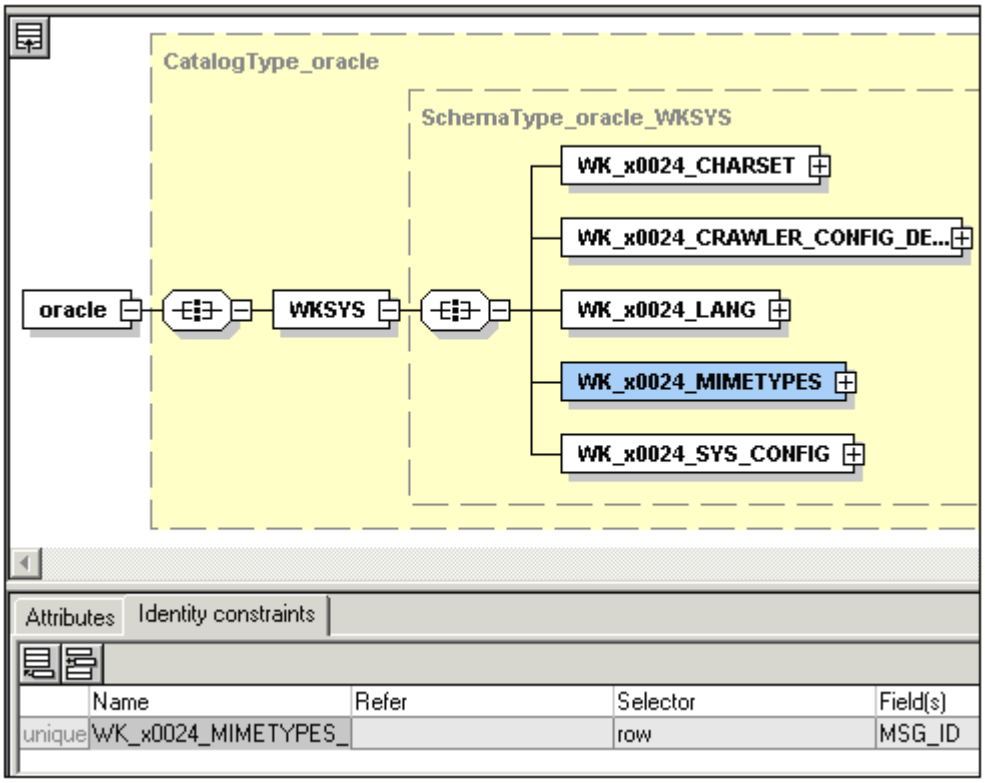
© 2005 Altova GmbH



7. Klicken Sie auf das Komponentensymbol , um das Inhaltsmodell dieser globalen Komponente zu sehen.
   Klicken Sie auf das Register "**Identity constraints**", um die keyref- und key-Felder der entsprechenden Elemente anzuzeigen.

   Hinweis: Die Oracle Servererweiterungen stehen in der Eingabehilfe Details auf dem Register **Oracle** zur Verfügung.

Ergebnis nach Generierung eines flachen Schemamodells:
Wenn Sie im Dialogfeld "Schema erzeugen" die Option "Flach (SQL/XML-Standard)" gewählt haben, sieht das erzeugte Schema wie unten gezeigt aus.



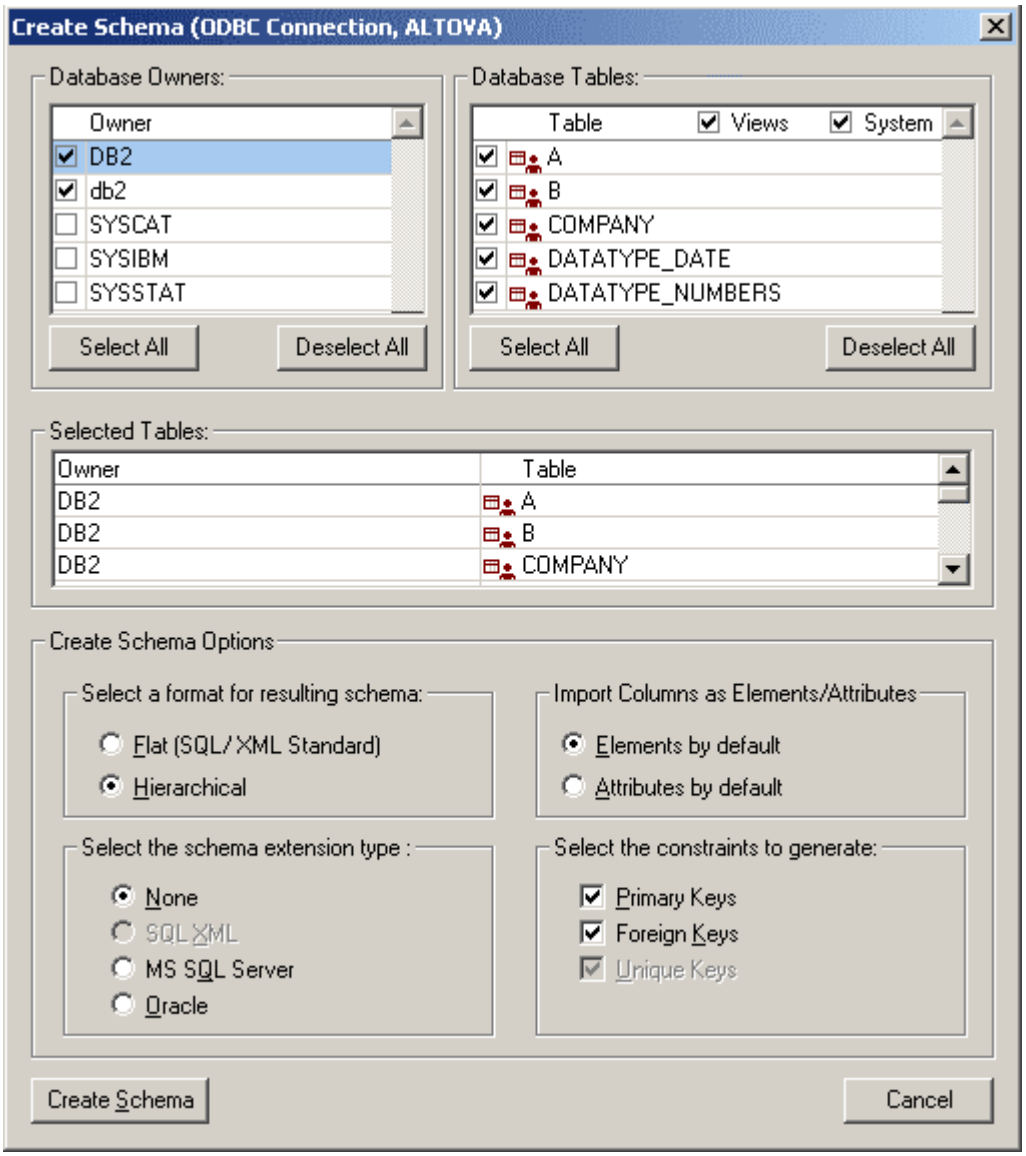
XML-Schema von DB-Struktur erzeugen (IBM DB2)

Um das folgende Beispiel auf Ihrem Rechner durchgehen zu können, muss auf Ihrem PC bzw. Netzwerk der IBM DB2 Client installiert sein.

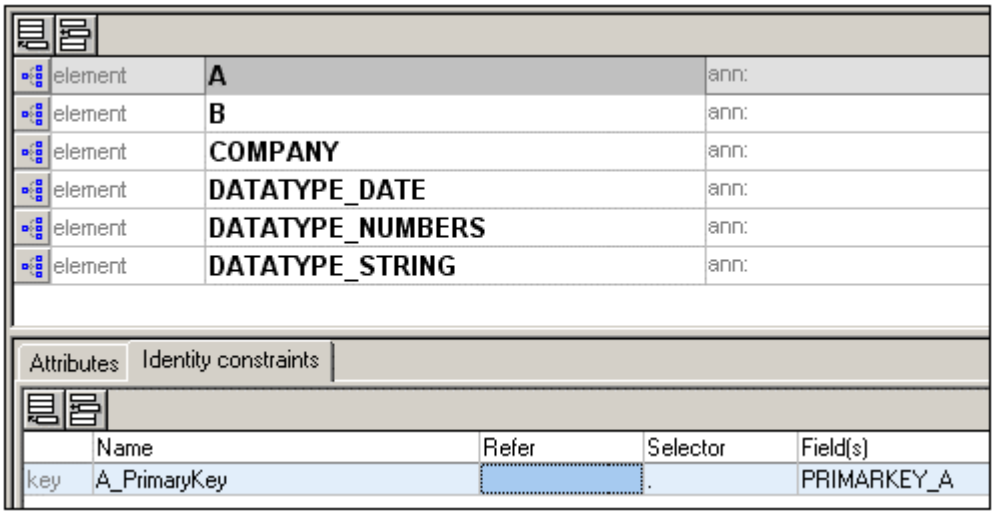
So erzeugen Sie ein hierarchisches Schema von einer Datenbankdatei (IBM DB2):

1. Wählen Sie die Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen.**
2. Wählen Sie **IBM DB2 (ODBC)** und klicken Sie auf die Schaltfläche **Weiter**.
   Das Dialogfeld "**IBM DB2 : ODBC Connection String**" wird geöffnet.
3. Klicken Sie auf die Schaltfläche **Erzeugen,** um die Datenbankverbindungseinstellungen auszuwählen.
   Daraufhin wird das Dialogfeld "**Datenquelle aussuchen**" geöffnet.
3. Klicken Sie auf das Register "**Computerdatenquelle"**, um den Namen der Datenquelle auszuwählen, der erstellt wurde, als der IBM DB2 Client installiert wurde (Altova), und klicken Sie auf OK.
   Das Dialogfeld "**Mit DB2-Datenbank verbinden**" wird geöffnet.
4. Geben Sie die **User ID** und das **Passwort** ein und klicken Sie auf OK, um die Verbindung zur Datenbank herzustellen.
   Der Connection String wird erstellt und im Textfeld "Connection String" eingesetzt.
5. Klicken Sie zum Fortfahren auf Weiter.
   Daraufhin wird das Dialogfeld "**Datenbank importieren**" geöffnet.

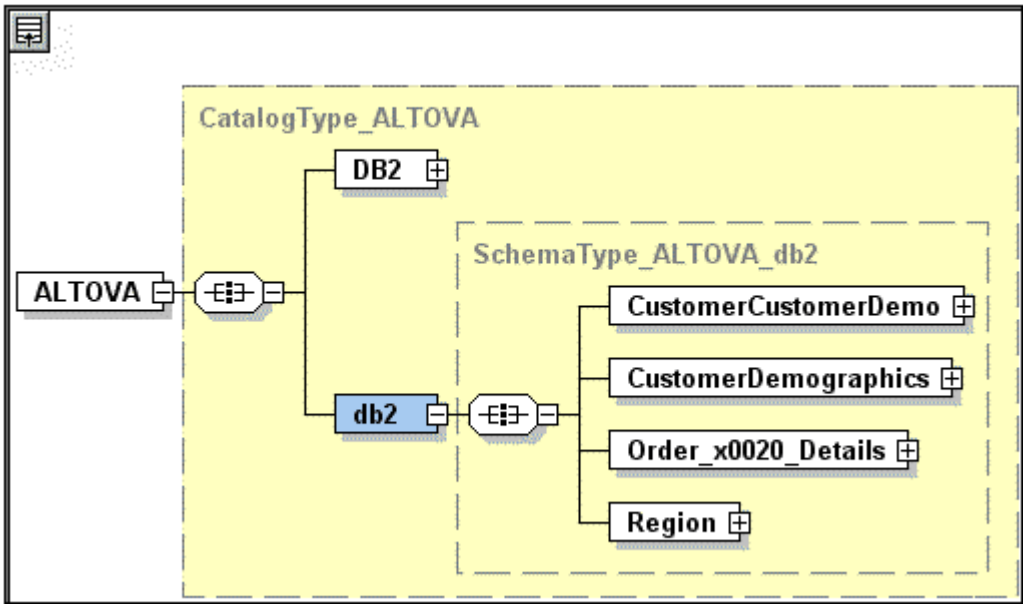
© 2005 Altova GmbH



6. Klicken Sie im Listenfeld "Datenbank-Owner" auf den **Owner** und wählen Sie die Tabellen aus, die im Schema aufscheinen sollen.
7. Klicken Sie auf die Schaltfläche "**Schema erzeugen"**, um die Konvertierung zu starten.
8. Klicken Sie auf das Komponentensymbol , um das Inhaltsmodell dieser globalen Komponente zu sehen.



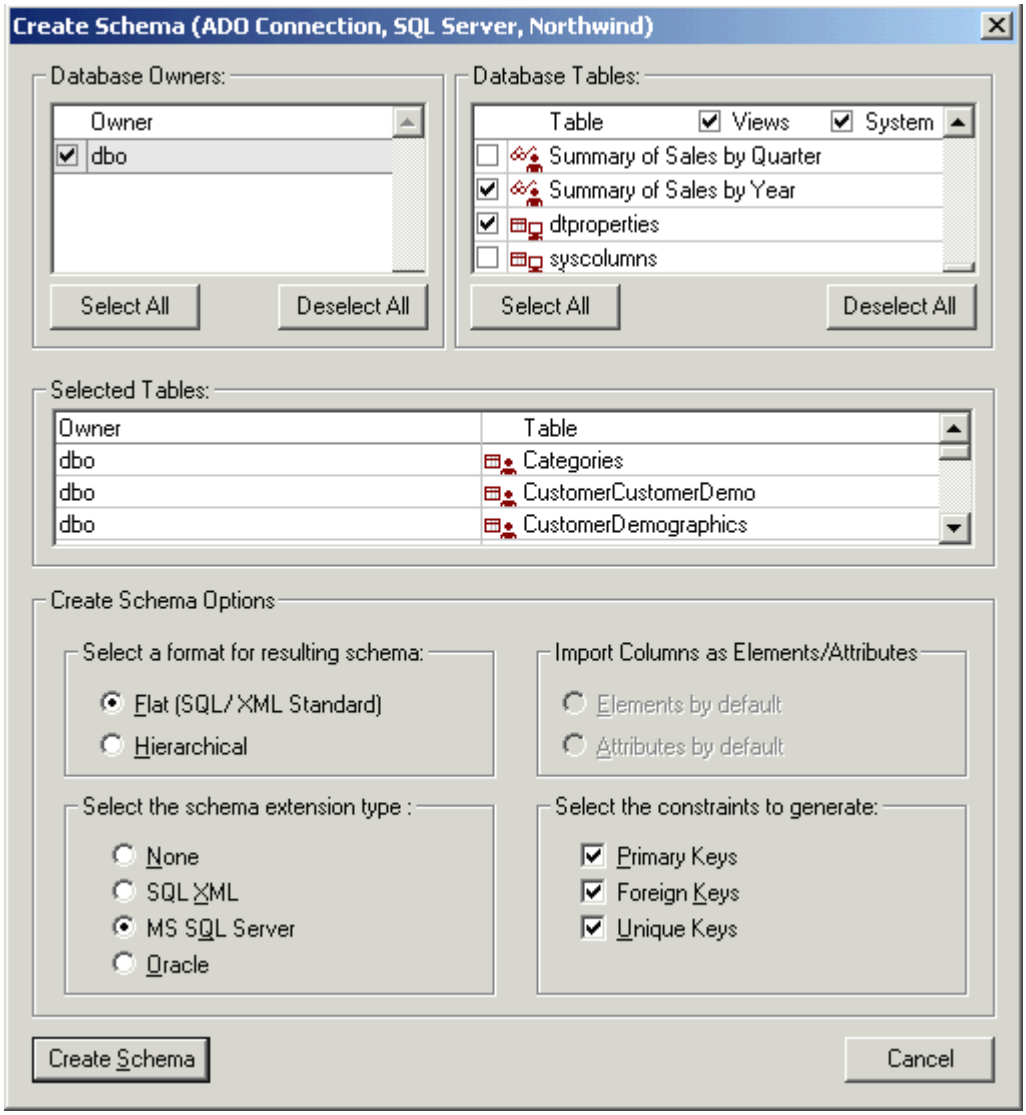
Ergebnis nach Generierung eines flachen Schemamodells:
Wenn Sie im Dialogfeld "Schema erzeugen" die Option "Flach (SQL/XML-Standard)" gewählt haben, sieht das erzeugte Schema wie unten gezeigt aus.



 © 2005 Altova GmbH

Optionen zum Erzeugen von XML-Schemas

XML-Schemaformate:



- Im **hierarchischen** Schemamodell werden Tabellenabhängigkeiten grafisch in Form einer Baumdiagrammansicht dargestellt, in der abhängige Tabellen im Inhaltsmodell als eingerückte Child-Elemente dargestellt werden. Die Tabellenabhängigkeiten werden auch auf dem Register "Identity Constraints" angezeigt.

  Tabellen werden im Schema als globale Elemente aufgelistet, während Spalten die Elemente bzw. Attribute dieser globalen Elemente sind (Der Benutzer entscheidet, ob die Spalten als Elemente oder Attribute gemappt werden sollen). Die Beziehungen werden hierarchisch erstellt, so dass ein Sekundärschlüsselfeld in einer Tabelle auf das globale Element verweist, das die Tabelle darstellt.

- Das **flache** Schemamodell basiert auf einem ISO-ANSI Working Draft mit dem Titel XML-Related Specification (SQL/XML) (in Zukunft als " SQL/XML Working Draft" bezeichnet). Im SQL/XML Working Draft ist definiert, wie Datenbanken auf XML gemappt werden.

  Beziehungen werden im Schema mittels Identity Constraints definiert. Es gibt keine

Referenzen auf Elemente. Das Schema weist daher eine flache Struktur auf, die der Baumstruktur in der Datenbank ähnelt. Nähere Informationen zum SQL/XML Working Draft finden Sie unter: http://www.sqlx.org/SQLXdocs

Null Constraints und Schema-Standardwerte
Wenn die Spalte den Constraint NULL hat, wird den Elementen das Attribut "Nillable=true" und den Attributen "Use=optional" angehängt. Standardwerte werden nur von Spalten vom Typ Zahl oder String erzeugt.

Wenn Spalten als Attribute im hierarchischen Format generiert werden, gibt es noch eine zusätzliche Einschränkung. Ist ein Standardwert definiert und das Attribut "User" "required", wird der Standardwert nicht generiert.

Tabellen und Ansichten
Es können die folgenden vier Tabellentypen aus einer Datenbank abgerufen werden:

- Benutzertabellen
- Systemtabellen
- Benutzeransichten
- Systemansichten

Die verschiedenen Tabellentypen werden durch Symbole neben den ausgewählten Tabellen- und Owner-Namen gekennzeichnet (falls sie in der Datenbank verfügbar sind).

Bitte beachten Sie, dass die Unterscheidung zwischen Benutzer und System bei einer Verbindung über ODBC (nur bei Ansichten) verloren geht. Alle Tabellen sind daher Benutzertabellen und alle Ansichten Benutzeransichten.

MS Access Systemtabellen werden nicht angezeigt, da es im Moment nicht möglich ist, diese Tabellen abzurufen.

Owner
Nicht alle Datenbanken unterstützen das Konzept Owner. So gibt es z.B. in MS Access keine Owner einer Datenbank. In diesem Fall werden nur die Tabellen, die in der Datenbank gefunden werden, aufgelistet und generiert. Daher wird das "flache" Schemaformat nicht unterstützt.

Identity Constraints und Beziehungen
Identity constraints und Beziehungen werden mittels ADO-, ODBC- und OCI-Importmethoden abgerufen bzw. generiert.

Beim Import einer MS Access-Datenbank über ODBC werden keine Identity Constraints abgerufen, da MS Access diese Funktion über ODBC nicht unterstützt.

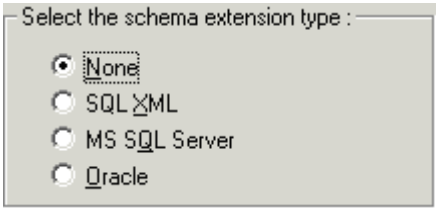
Wenn Sie in MySQL mit Sekundärschlüsseln arbeiten, beachten Sie, dass die erzeugten Tabellen vom Typ INNODB sein müssen (die jeweils verwendete MySQL-Version muss diesen Typus ebenfalls unterstützen). Dies ist eine Einschränkung, die für MySQL-Datenbanken gilt. Nähere Informationen dazu finden Sie unter:
http://www.mysql.com/doc/en/InnoDB_foreign_key_constraints.html

Eindeutige Schlüssel werden derzeit über ODBC nicht abgerufen/unterstützt.

© 2005 Altova GmbH

Schema-Extensions erzeugen

Schema-Extensions



Schema-Extension-Informationen sind zusätzliche Informationen, die aus einer Datenbank ausgelesen werden und anschließend in einer der folgenden Formen in das Schema eingebettet werden:

- als Annotationsdaten oder
- als Attribute

Beim Generieren von Schemas stehen vier Möglichkeiten zur Auswahl:

- Keine Extension-Informationen
- SQL-Server-Extensions
- Oracle-Extensions
- SQL/XML-Extensions

Keine:
Bei Auswahl von "Keine" liefert die Datenbank keine zusätzlichen Informationen.

SQL-Server Extensions:
Bei Auswahl der Option Microsoft SQL Server werden SQL-Server-Extensions generiert. Nähere Informationen zu diesem Thema finden Sie online in den SQL Server Books (Aktualisiert -SP3) unter "Using Annotations in XSL Schemas" (Download von: http://www.microsoft.com/sql/techinfo/productdoc/2000/books.asp).

Im Schema werden die folgende Annotationsdaten-Untermengen generiert:

- sql:relation
- sql:field
- sql:datatype
- sql:mapped

Oracle:
Wenn Sie mit einer Oracle-Datenbank arbeiten, werden standardmäßig Oracle-Extensions ausgewählt. Zusätzliche Datenbankinformationen werden als Attribute gespeichert. Eine ausführliche Beschreibung dieser Attribute finden Sie unter http://download-west.oracle.com/docs/cd/B10501_01/appdev.920/a96620/xdb05obj.htm#1027227 (Sie müssen Zugriff auf das Oracle Technology Network haben, um diese Seite aufrufen zu können.)

Zur Zeit wird die folgende Attributuntermenge generiert:

- SQLName
- SQLType
- SQLSchema

SQL/XML
SQL/XML-Erweiterungen werden nur bei der Erzeugung von Schemas im flachen Format eingefügt. Die Extension-Informationen werden in Form von Annotationen gespeichert. Eine nähere Beschreibung finden Sie im SQL/XML Working Draft.

Auch wenn Oracle- und SQL-Servererweiterungen für den jeweiligen Datenbanktyp generiert werden, besteht in dieser Hinsicht keine Einschränkung. Dies erweist sich dann als nützlich, wenn Sie mit einer dritten Datenbank arbeiten und ein Schema erzeugt werden soll, das später mit Oracle bzw. SQL Server kompatibel sein soll.

Mappen auf XML-Schema-Datentypen

In den folgenden Tabellen sind die einzelnen Mappings für die jeweiligen XML-Schema-Datentypen aufgelistet.

XML-Datentypen und MS SQL Server

SQLServer	**XMLDataType**
	ID
char	String
nchar	
varchar	
nvarchar	
text	
ntext	
sysname	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
bit	Boolean
real	Float

© 2005 Altova GmbH

float	Double
decimal	Decimal
money	
smallmoney	
	Duration
datetime	Datetime
smalldatetime	
	Time
	Date
	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
binary	Bin_base64
varbinary	
image	
	Bin_hex
integer	Integer
	Int
	NegativeInteger
	PositiveInteger
	NonNegativeInteger
	NonPositiveInteger
	UnsignedInt
smallint	Short
	UnsignedShort
bigint	Long
	UnsignedLong
tinyint	Byte
	UnsignedByte

XML-Datentypen und MS Access

MSAccess	**XMLDataType**
GUID	ID
char	String
varchar	
memo	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
bit	Boolean
Number(single)	Float
Number(double)	Double
Decimal	Decimal
Currency	
Currency	
	Duration
Date/Time	Datetime
Date/Time	
	Time
	Date
	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
	Bin_base64
(noequivalent)	
OLEObeject(image)	
	Bin_hex
Number(LongInteger)	Integer
	Int

© 2005 Altova GmbH

	NegativeInteger
	PositiveInteger
	NonNegativeInteger
	NonPositiveInteger
	UnsignedInt
Number(Integer)	Short
	UnsignedShort
(noequivalent)	Long
	UnsignedLong
Number(Byte)	Byte
	UnsignedByte

XML-Datentypen und Oracle

Oracle	**XMLDataType**
ROWID	ID
CHAR	String
NCHAR	
VARCHAR2	
NVARCHAR2	
CLOB	
NCLOB	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
NUMBER	Boolean
Constraintapplied	
FLOAT	Float
FLOAT	Double

NUMBER	Decimal
NUMBER	
NUMBER	
	Duration
DATE	Datetime
DATE	
	Time
DATE	Date
INTERVALYEARTOMONTH	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
BLOB	Bin_base64
BLOB	
BLOB	
	Bin_hex
NUMBER	Integer
	Int
	NegativeInteger
	PositiveInteger
	NonNegativeInteger
	NonPositiveInteger
	UnsignedInt
NUMBER	Short
	UnsignedShort
NUMBER	Long
	UnsignedLong
NUMBER	Byte
	UnsignedByte

XML-Datentypen und MySQL

© 2005 Altova GmbH

MySQL	**XMLDataType**
	ID
char	String
varchar	
text,tinytext,mediumtext	
longtext	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
tinyint(1)	Boolean
float	Float
double	Double
decimal	Decimal
decimal	
decimal	
	Duration
datetime	Datetime
datetime	
	Time
	Date
	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
blob,tinyblob,mediumblob,longblob	Bin_base64
	Bin_hex
	Integer
	Int
	NegativeInteger
	PositiveInteger
	NonNegativeInteger

	NonPositiveInteger
	UnsignedInt
smallint	Short
	UnsignedShort
bigint	Long
	UnsignedLong
tinyint	Byte
	UnsignedByte

XML-Datentypen und Sybase

Sybase	**XMLDataType**
	ID
char	String
nchar	
varchar	
nvarchar	
text	
sysname-varchar(30)	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
bit	Boolean
real,float	Float
double,float	Double

© 2005 Altova GmbH

decimal	Decimal
money	
smallmoney	
	Duration
datetime	Datetime
smalldatetime	
	Time
	Date
timestamp	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
binary<=255	Bin_base64
varbinary<=255	
image	
	Bin_hex
integer	Integer
	Int
	NegativeInteger
	PositiveInteger
	NonNegativeInteger
	NonPositiveInteger
	UnsignedInt
smallint	Short
	UnsignedShort
(noequivalent)	Long
	UnsignedLong
tinyint	Byte
	UnsignedByte

XML-Datentypen und ODBC

ODBC	**XMLDataType**
SQL_GUID	ID
SQL_CHAR	String
SQL_VARCHAR	
SQL_LONGVARCHAR	
SQL_LONGVARCHAR	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
SQL_BIT	Boolean
SQL_REAL	Float
SQL_DOUBLE	Double
SQL_DECIMAL	Decimal
SQL_DECIMAL	
SQL_DECIMAL	
	Duration
SQL_TIMESTAMP	Datetime
SQL_TIMESTAMP	
	Time
SQL_DATE	Date
SQL_TIMESTAMP	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
SQL_BINARY	Bin_base64
SQL_VARBINARY	
SQL_LONGVARBINARY	
	Bin_hex
SQL_INTEGER	Integer
	Int
	NegativeInteger
	PositiveInteger
	NonNegativeInteger

© 2005 Altova GmbH

	NonPositiveInteger
	UnsignedInt
SQL_SMALLINT	Short
	UnsignedShort
SQL_BIGINT	Long
	UnsignedLong
SQL_TINYINT	Byte
	UnsignedByte

XML-Datentypen und ADO

ADO	**XMLDataType**
adGUID	ID
adChar	String
adWChar	
adVarChar	
adWVarChar	
adLongVarChar	
adWLongVarChar	
adVarWChar	
	NormalizedString
	Token
	Name
	NCName
	anyURi
	QName
	Notation
adBoolean	Boolean
adSingle	Float
adDouble	Double

adNumeric	Decimal
adCurrency	
adCurrency	
	Duration
adDBTimeStamp	Datetime
adDBTimeStamp	
	Time
adDate	Date
adDBTimeStamp	gYearMonth
	gYear
	gMonthDay
	gDay
	gMonth
adBinary	Bin_base64
adVarBinary	
adLongVarBinary	
	Bin_hex
adInteger	Integer
	Int
	NegativeInteger
	PositiveInteger
	NonNegativeInteger
	NonPositiveInteger
adUnsignedInt	UnsignedInt
adSmallInt	Short
adUnsignedSmallInt	UnsignedShort
adBigInt	Long
adUnsignedBigInt	UnsignedLong
adTinyInt	Byte
adUnsignedTinyInt	UnsignedByte

13.9.5 DB-Import basierend auf XML-Schema

Mit XMLSpy 2005 können Sie XML-Dokumente erstellen, die Datenbankendaten enthalten, welche auf existierenden XML-Schemas basieren. Das Schema wird über die Menüoption **Konvertieren | XML-Schema von DB-Struktur erzeugen** erstellt.

Die folgenden Datenbanken werden unterstützt:

- Microsoft Access 2000 und 2003
- Microsoft SQL Server

© 2005 Altova GmbH

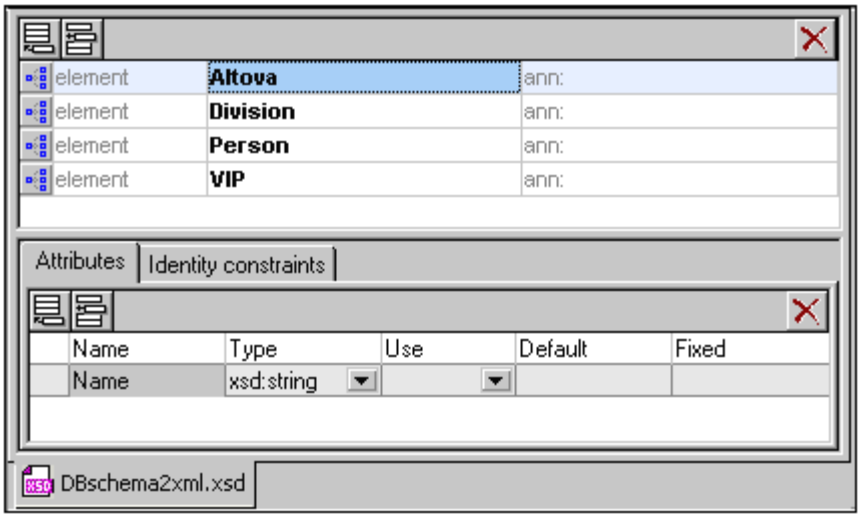
- Oracle
- MySQL
- Sybase
- IBM DB2

Als Quelldatenbank kann eine Microsoft Access-Datenbank verwendet werden. Sie können auch einen ADO Connection String erzeugen, um eine Verbindung zur Datenbank Ihrer Wahl herzustellen. Beachten Sie, dass Sie mit Hilfe dieser Option nur ADO-Verbindungen zu einer dieser Datenbanken erzeugen können. Wenn Sie mit Oracle, MySQL, Sybase und IBM DB2 arbeiten, wird empfohlen, die Verbindung zur jeweiligen Datenbank mit Hilfe des MS OLE-DB Provider für ODBC-Treiber zu erstellen.

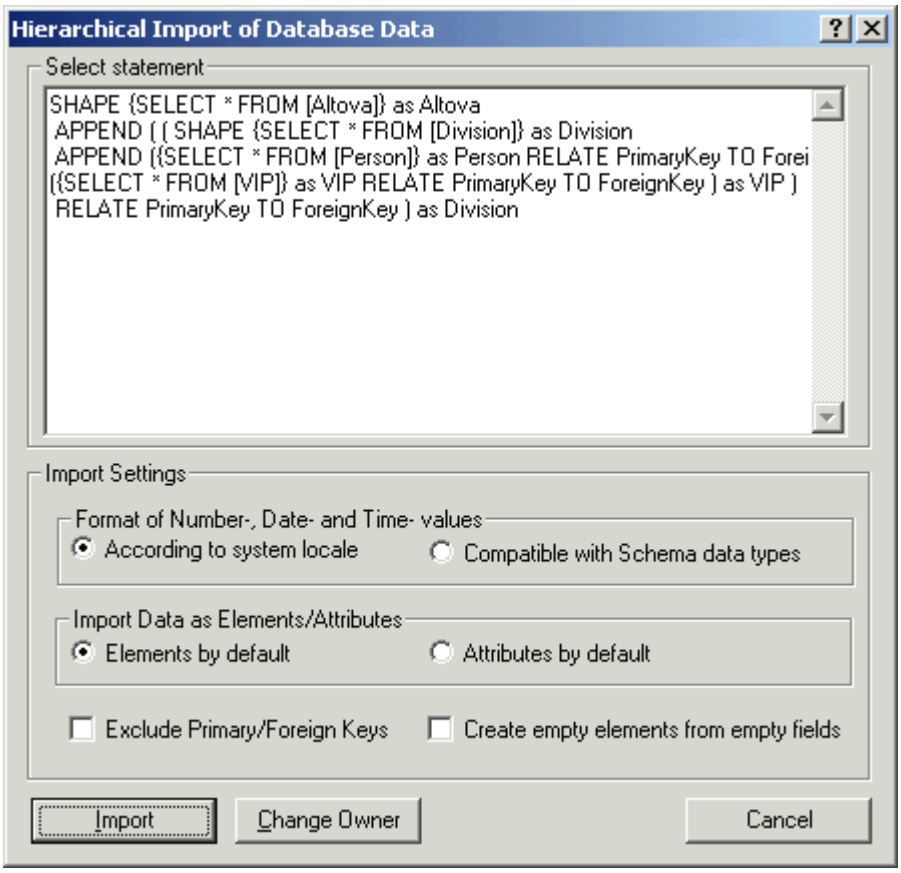
Festlegen der Daten, die importiert werden sollen
Sie können den zu importierenden Datensatz selbst festlegen, indem Sie angeben, welche Tabelle als Root-Tabelle des zu importierenden Datensatzes fungieren soll. Es wird dann automatisch ein Shapestring erstellt, der im Textfeld Selec-Anweisung angezeigt wird. Die Datenbankdaten werden dann mit allen hierarchischen Beziehungen importiert.

So importieren Sie XML-Daten aus einer Microsoft Access Datenbank:

1. Öffnen Sie die Schemadatei **DBschema2xml.xsd**, die in XMLSpy 2005 zur Verfügung steht (Ordner Tutorial).

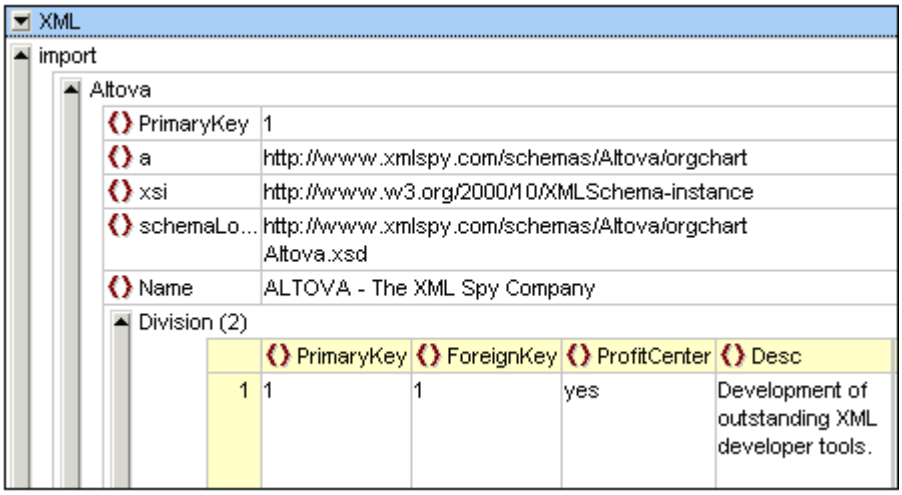


2. Wählen Sie die Menüoption **Konvertieren | DB-Import auf Basis von XML-Schema**. Das Dialogfeld "**Quelldatenbank auswählen**" wird geöffnet. Wählen Sie die Option "**Microsoft Access (ADO)**" und klicken Sie auf die Schaltfläche "Weiter". Klicken Sie auf die Schaltfläche Durchsuchen, um eine bestehende MS Access-Datenbank (**DB2schema.mdb**) auszuwählen und klicken Sie auf Weiter.
3. Wählen Sie im Dialogfeld eine Root-Tabelle aus, z.B. **Altova**.
   Hier definieren Sie, welche Tabellen importiert werden sollen (in diesem Fall Altova). Die Root-Tabelle wird in der importierten Tabellenhierarchie an höchster Stelle stehen.
4. Nachdem Sie die Root-Tabelle ausgewählt haben, bestätigen Sie Ihre Auswahl mit **OK**.



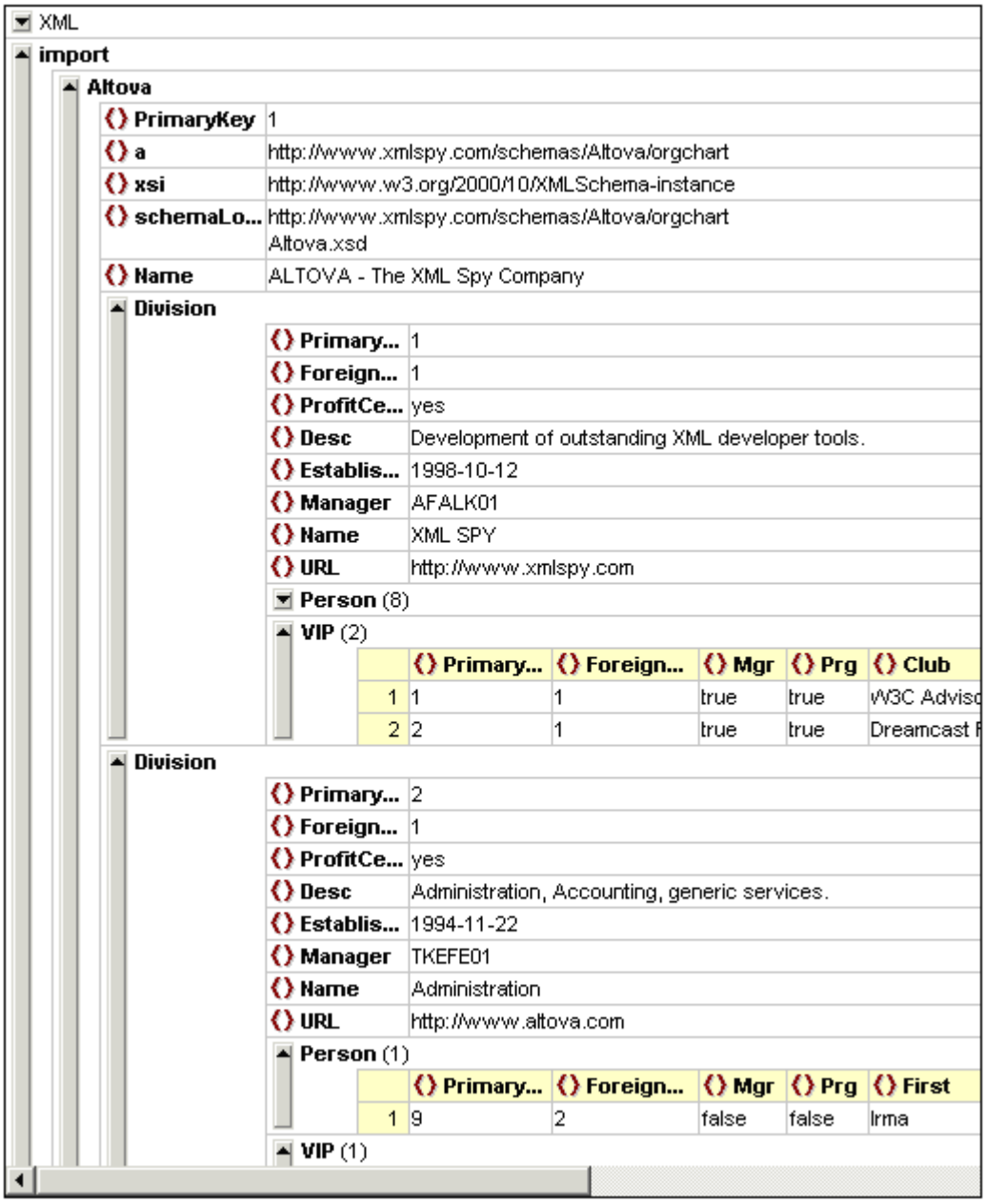
Die Select-Anweisung, basierend auf Ihrer Root-Element Auswahl, wird automatisch erzeugt und im Textfeld "Select-Anweisung" angezeigt. Sie können die Select-Anweisung bei Bedarf editieren, um die Importparameter zu ändern.

5. Klicken Sie auf die Schaltfläche **Import,** um die Datenbankdaten zu importieren. Das XML-Dokument wird als eine "Untitledx.xml"-Datei im Hauptfenster angezeigt.



6. Klicken Sie auf das Element Altova und wählen Sie die Menüoption **XML | Tabelle | Als Tabelle anzeigen**, um die Tabellenformatierung zu entfernen und die Daten vertikal anzuzeigen. Die Elemente Division wurden im unten angezeigten Diagramm gleichermaßen behandelt.

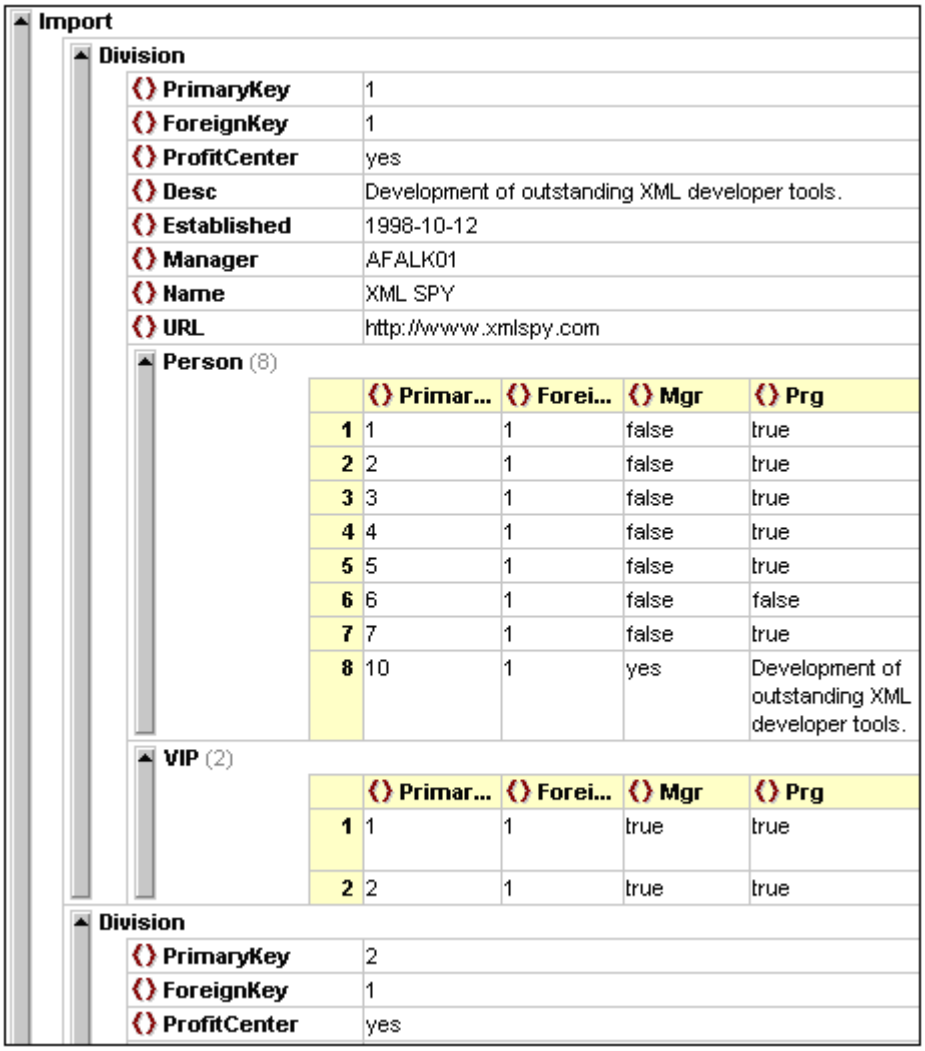
© 2005 Altova GmbH



In diesem Fall wurde die gesamte Datenbank importiert, wobei Altova die Root-Tabelle bildet. Das Datenelement Altova Name wurde gemäß der Schemadefinition DBschema2xml.xsd (am Beginn des Abschnitts auf dem Register Attribut des Schemas) als Attribut importiert.

Partiellen Datensatz importieren:

1. Wiederholen Sie die obigen Schritte, und wählen Sie **Division** als Root Tabelle.
2. Verwenden Sie als Datenquelle die Datei **DB2schema.mdb**.
   Das Diagramm zeigt das resultierende XML-Dokument. Die Tabelle Person ist erweitert und in der Tabelle Division wurde für einen besseren Überblick die Formatierung deaktiviert.



Import

Division

PrimaryKey	1
ForeignKey	1
ProfitCenter	yes
Desc	Development of outstanding XML developer tools.
Established	1998-10-12
Manager	AFALK01
Name	XML SPY
URL	http://www.xmlspy.com

Person (8)

	Primar...	Forei...	Mgr	Prg
1	1	1	false	true
2	2	1	false	true
3	3	1	false	true
4	4	1	false	true
5	5	1	false	true
6	6	1	false	false
7	7	1	false	true
8	10	1	yes	Development of outstanding XML developer tools.

VIP (2)

	Primar...	Forei...	Mgr	Prg
1	1	1	true	true
2	2	1	true	true

Division

PrimaryKey	2
ForeignKey	1
ProfitCenter	yes

Problem beim Verbinden mit MySQL:

Verwenden Sie bitte beim Erstellen einer ADO-Verbindung auf Basis von ODBC keine Datei-DSN. Die Datei-DSN verursacht die Fehlermeldung "Klasse nicht registriert"; verwenden Sie stattdessen entweder die Benutzer- oder System-DSN.

13.9.6 DB-Struktur von XML-Schema erzeugen

Mit XMLSpy 2005 können Sie eine leere Datenbank (oder ein Datenbankgerüst) basierend auf einer existierenden Schemadatei erzeugen. Die Schemastruktur, die durch die Identity Constraints definiert wird, wird in die neue Datenbank übernommen.

In der unten gezeigten Tabelle sehen Sie die erzeugten Datenbanktypen, die Einschränkungen und die Verbindungsmethoden, wenn Sie die Menüoption "XML-Schema von DB-Struktur erzeugen" verwenden.

© 2005 Altova GmbH

	DB-Struktur von XML-Schema erzeugen				
	Direkte	über ODBC	über ADO	Unterstützung für Identity Constraint	Flaches oder hierarchisches DB-Format
Microsoft Access	OK *	OK	OK		
MS SQL Server	OK *	OK	OK		
Oracle	OK *	OK	OK		
MySQL	-	OK *	OK W		
Sybase	-	OK *	OK		
IBM DB2	-	OK *	OK		

***** **Empfohlene Verbindungsmethode für die jeweilige Datenbank.**

W **MySQL: Bei Herstellen der ADO-Verbindung auf Basis von ODBC wird empfohlen, entweder die Benutzer- oder die System-DSN zu verwenden.**

- **Nicht unterstützt**

XMLSpy 2005 kann sowohl Schemas in hierarchischem als auch Schemas in flachem Format mappen. XMLSpy 2005 erkennt beide Formate automatisch.
Das flache Format kann auf zwei Arten auf SQL gemappt werden.

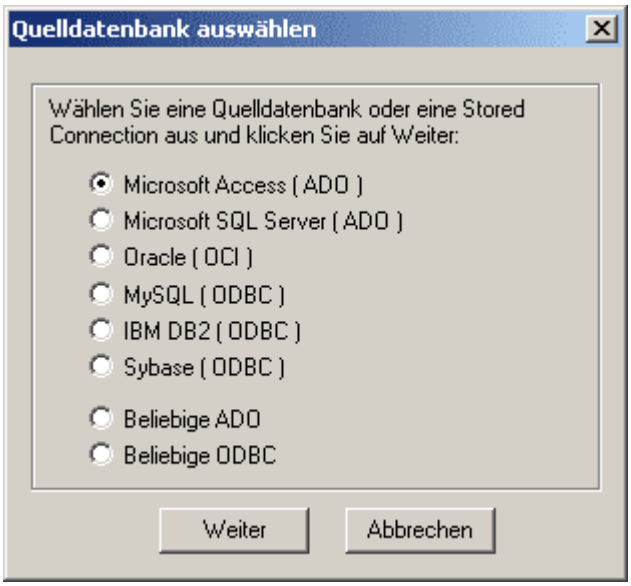
- SQL Server DB, Oracle DB oder Sybase DB:
  Bei einem Schema, das für eine der oben genannten Datenbanken im flachen Format generiert wurde, wird der Schema-Katalogname extrahiert und im allgemeinen SQL-Skript als DB-Name verwendet. Das bedeutet, dass das erzeugte SQL-Skript an einer Zieldatenbank ausgeführt wird, deren Name mit dem Schemakatalognamen identisch sein muss.

- ACCESS, MySQL oder DB2 DB:
  Bei einem Schema, das für eine der oben genannten Datenbanken im flachen Format generiert wurde, wird der Schemakatalogname **ignoriert**, wenn ein SQL-Skript erstellt wird. Das bedeutet, dass das erzeugte SQL-Skript an einer Zieldatenbank ausgeführt wird, an der ein Login erfolgt ist.

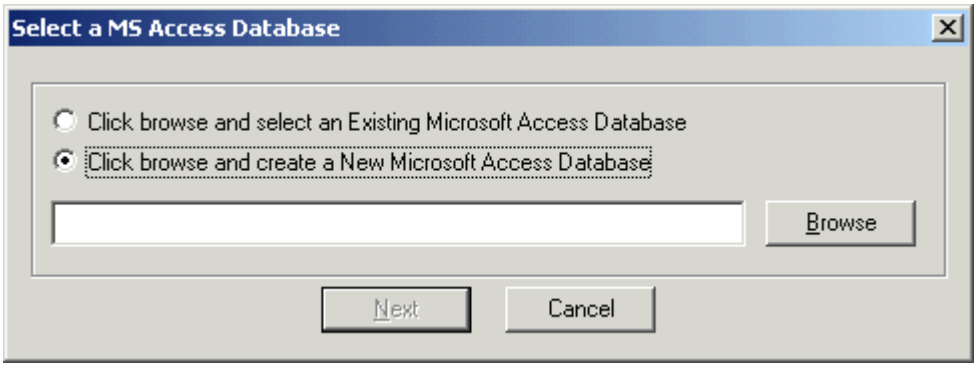
Die unten beschriebene Methode ist im Allgemeinen bei jedem Datenbanktyp dieselbe. Spezifische Informationen dazu, wie datenbankspezifische Datentypen auf einzelne Schemadatentypen gemappt werden, finden Sie im Abschnitt "Mappen von XML-Schemadatentypen".

So erzeugen Sie eine Datenbank auf Basis eines Schemas:

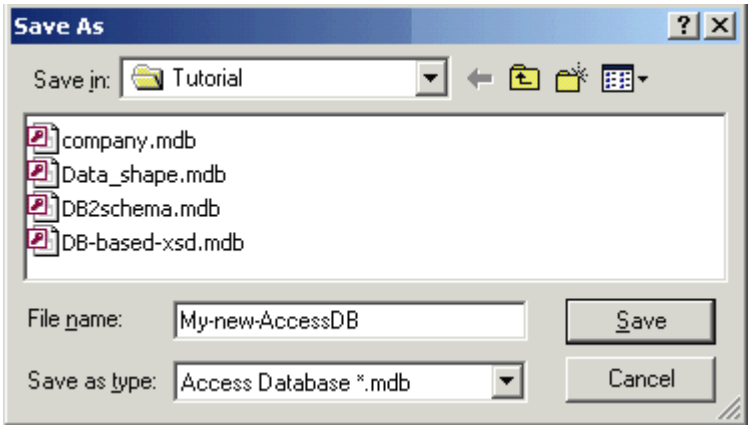
1. Öffnen Sie die Schemadatei **No-constraints.xsd** im Ordner Tutorial.
2. Wählen Sie die Menüoption **Konvertieren | DB-Struktur von XML-Schema erzeugen**.
3. Klicken Sie auf das Optionsfeld "**Microsoft Access**" und anschließend auf **Weiter**.



Daraufhin wird das Dialogfeld **MS Access-Datenbank auswählen** geöffnet. Sie können nun angeben, ob Sie eine bestehende Datenbank überschreiben oder eine neue erstellen wollen.



4. Wählen Sie die Option "**Klicken Sie auf Durchsuchen und erstellen Sie...**" und anschließend auf die Schaltfläche "Durchsuchen".



5. Geben Sie den Namen der Datenbank ein, die erstellt werden soll (z.B. My-new AccessDB) und klicken Sie auf Speichern.
Daraufhin wird wieder das vorhergehende Dialogfeld angezeigt. Im Textfeld sind der

 © 2005 Altova GmbH

Name und Pfad der Datenbank eingetragen.

6. Klicken Sie zum Fortfahren auf die Schaltfläche **Weiter**.
   Wenn das Schema, das Sie als Basis verwenden, **keine** vordefinierten Identity Constraints aufweist, wird das Dialogfeld **Beziehungen definieren** geöffnet.

   Die in diesem Beispiel definierte Datei hat keine vordefinierten Identity Constraints, daher müssen die Beziehungen manuell definiert werden. (Beachten Sie, dass sich dies durch Auswahl von **Konvertieren | XML-Schema von Datenbank erzeugen** und **Deaktivieren** aller drei **Identity Constraint-**Kontrollkästchen einstellen lässt.)

7. Wählen Sie die erste Zeile aus und klicken Sie auf die Schaltfläche **Schlüssel definieren** (oder doppelklicken Sie auf eine Zeile).

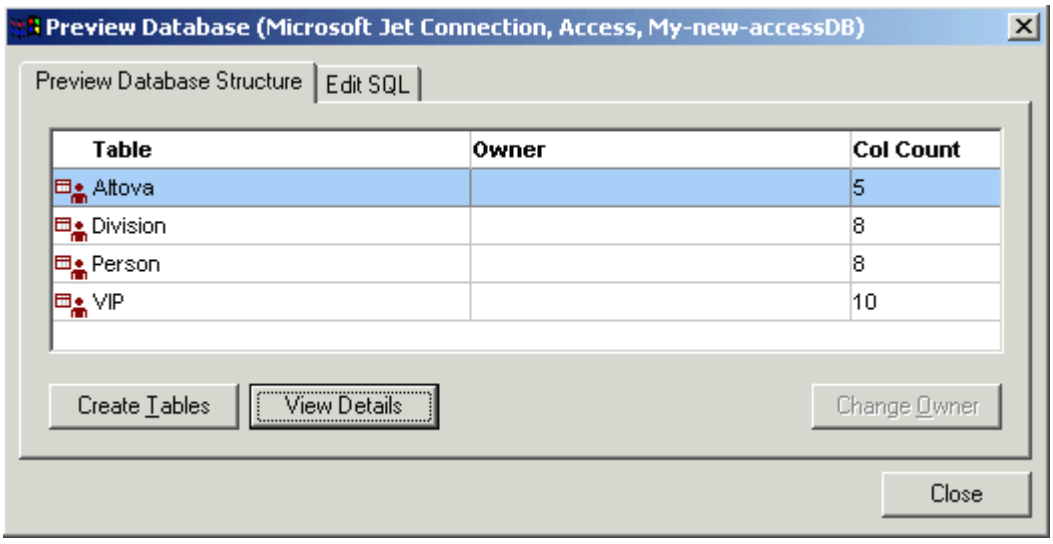
   In der ersten Spalte können Sie das Primärschlüsselfeld der Tabelle "Altova" definieren.

8. Klicken Sie auf das Listenfeld und wählen Sie das Primärschlüsselfeld aus, in diesem Fall "PrimaryKey".
   In der zweiten Spalte können Sie den Sekundärschlüssel der Tabelle "Division" definieren.
9. Klicken Sie auf das Listenfeld und wählen Sie das Sekundärschlüsselfeld aus, in diesem Fall "ForeignKey".

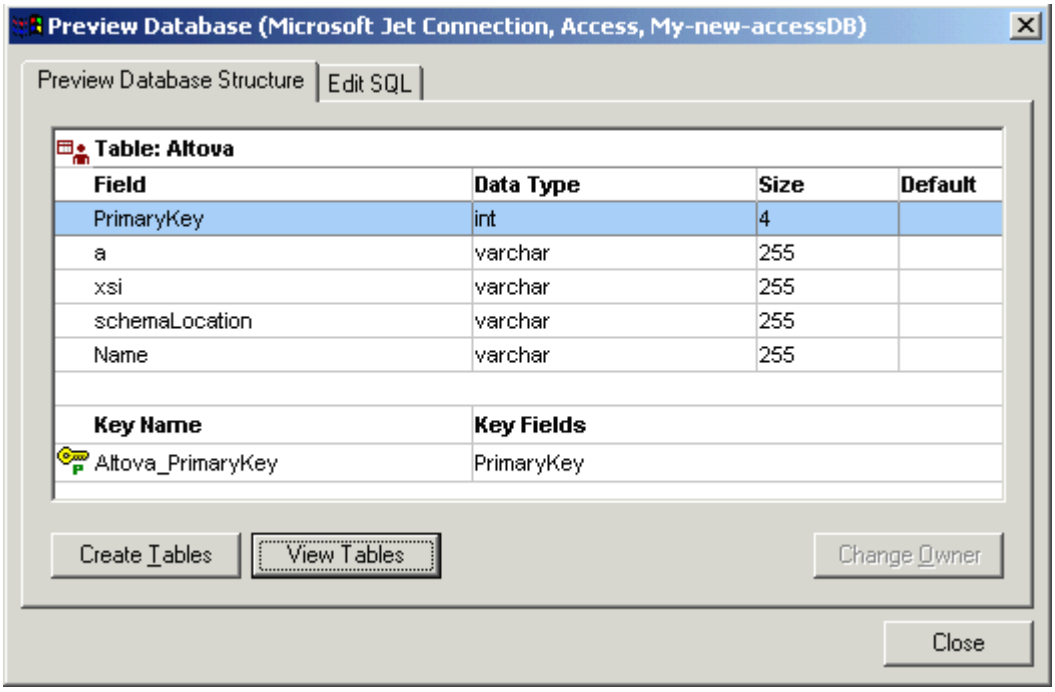
   Bitte beachten Sie:
   Beim Definieren einer spezifischen Bedingung ändert sich die Schaltfläche **Schlüssel definieren** in **List relationships.** Wenn Sie auf "List relationships" klicken, wird die Liste der Beziehungen angezeigt.

10. Definieren Sie die Beziehungen für die restlichen Tabellen und klicken Sie auf Weiter. Das Definieren von Beziehungen ist jedoch nicht obligatorisch.

    Das Dialogfeld **Datenbank-Vorschau** wird geöffnet. Auf dem Register **Datenbankstruktur-Vorschau** sehen Sie eine globale Übersicht der aktuellen Datenbankstruktur, während auf dem Register **SQL bearbeiten** die tatsächlich für das Erzeugen aller Datenbanktabellen verwendeten SQL-Befehle angezeigt werden.

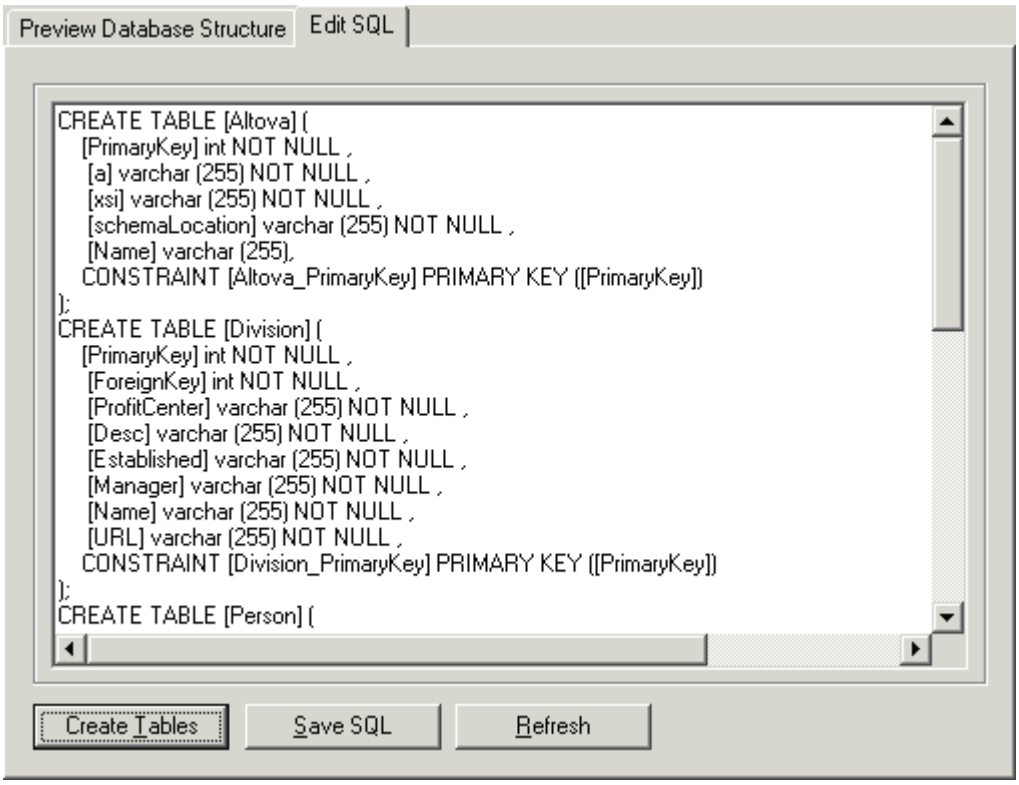


1. Klicken Sie auf die Schaltfläche **Details anzeigen,** um die Felddefinitionen der Tabelle Altova zu sehen.



2. Klicken Sie auf die Schaltfläche **View Tables,** um zur Tabellenübersicht zurückzugelangen.
3. Klicken Sie auf das Register **SQL bearbeiten,** um die aktuellen SQL-Anweisungen zu sehen und/oder zu editieren.

© 2005 Altova GmbH



Auf dem Register "SQL bearbeiten" können Sie die automatisch generierte SQL-Select-Anweisung ansehen und **editieren.** Über die Schaltfläche **SQL speichern** wird die SQL-Anweisung in einer Datei gespeichert, während Sie durch Klicken auf **Aktualisieren** alle auf diesem Register vorgenommenen Änderungen zurücksetzen.

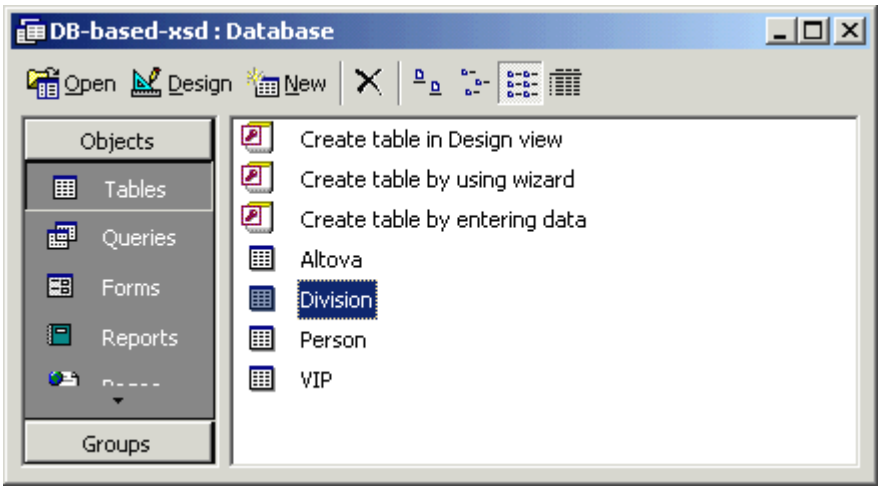
4. Klicken Sie auf die Schaltfläche **Tabellen erzeugen**, um die Access Datenbank mit den definierten Einstellungen zu generieren. Eine Meldung zeigt an, ob die Generierung erfolgreich war oder nicht.

Bitte beachten Sie:

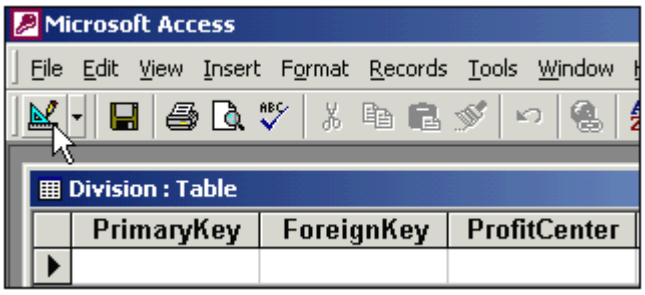
Änderungen, die auf diesem Register an der SQL-Anweisung vorgenommen wurden, werden verworfen, wenn Sie auf das Register "Datenbankstruktur-Vorschau" klicken. Wenn die Datenbanktabellen mit der geänderten SQL-Anweisung erzeugt werden sollen, stellen Sie sicher, dass Sie auf dem Register "SQL bearbeiten" auf die Schaltfläche **Tabellen erzeugen** klicken.

Anzeigen der generierten Datenbank:

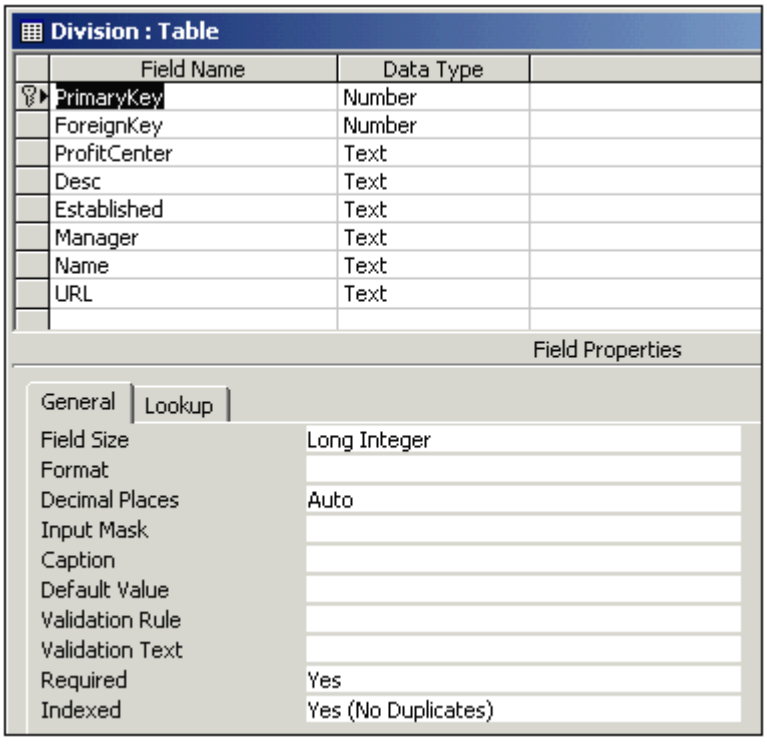
1. Doppelklicken Sie im Explorer auf die Datei **MY-new-accessDB.mdb** oder öffnen Sie sie in MS Access.



2. Doppelklicken Sie auf den Eintrag **Division,** um die Tabelle Division zu sehen.



3. Wählen Sie die Menüoption **Ansicht | Entwurfsansicht** oder klicken Sie auf das Symbol "Ansicht", um in die Design-Ansicht zu wechseln.



 © 2005 Altova GmbH

Die Design-Ansicht enthält Informationen zu den Feldeigenschaften der Datenbanktabelle: Feldnamen, Datentypen usw. Die hier angezeigten Parametereinstellungen wurden aus der Schemadefinition übertragen.

Mappen von XML-Schema-Datentypen

In den folgenden Tabellen sind die einzelnen Mappings für die jeweiligen Datenbanken aufgelistet. Die XML-Schema-Datentypen werden auf die einzelnen Datentypen der jeweiligen Datenbank gemappt.

XML-Datentypen und MS SQL Server

XML Data Type	**SQL Server**
ID	uniqueidentifier
String	If no facets if UNICODE nvarchar (255) else varchar (255) else if UNICODE Size = either length or maxlength If Size <= 4000 if FacetLengthIsSet nChar else nVarChar if Size <= 1073741823 nText else if NON UNICODE Size = either length or maxlength If Size <= 8000 if FacetLengthIsSet char

	else varchar if Size <= 2147483647 text
Normalized String	Refer to String
Token	Refer to String
Name	Refer to String
NCName	Refer to String
anyURi	Refer to String
QName	Refer to String
Notation	Refer to String
Boolean	bit
Float	real
Double	float
Decimal	decimal
Duration	datetime
Datetime	datetime
Time	datetime
Date	datetime
gYearMonth	datetime

© 2005 Altova GmbH

gYear	datetime
gMonthDay	datetime
gDay	datetime
gMonth	datetime
Bin_hex	If no facets varbinary (255) Size = either length or maxlength If Size <= 8000 if FacetLengthIsSet binary else varbinary if Size <= 2147483647 image
Bin_base64	Refer to Bin_hex
Integer	int
Int	int
Negative Integer	Int Constraint: restricted to the infinite set {...,-2,-1}
Positive Integer	Int Constraint: restricted to the infinite set {...,-2,-1}
Non Negative Integer	int Constraint: restricted to the infinite set {0, 1, 2, ...}
Non Positive Integer	int Constraint: restricted to the infinite set {...,-2 ,-1 ,0}
Unsigned Int	int Additional constraints
Short	smallint
Unsigned Short	smallint Additional constraints
Long	bigint
Unsigned Long	bigint Additional constraints
Byte	tinyint
Unsigned Byte	tinyint Additional constraints

XML-Datentypen und MS Access

XML Data Type	**MS Access**
ID	GUID
String	If no facets varchar (255) Size = either length or maxlength If Size <= 255 varchar (n) else memo
Normalized String	Refer to String
Token	Refer to String
Name	Refer to String
NCName	Refer to String
anyURi	Refer to String
QName	Refer to String
Notation	Refer to String
Boolean	bit
Float	Number (single)
Double	Number (double)
Decimal	Decimal
Duration	Date/Time

© 2005 Altova GmbH

Datetime	Date/Time
Time	Date/Time
Date	Date/Time
gYearMonth	Date/Time
gYear	Date/Time
gMonthDay	Date/Time
gDay	Date/Time
gMonth	Date/Time
Bin_hex	If no facets varbinary (255)
	Size = either length or maxlength
	If Size <= 8000 varbinary
	else image (OLE Object)
Bin_base64	
Integer	Number (Long Integer)
Int	Number (Long Integer)
Negative Integer	Number (Long Integer)
	Constraint: restricted to the infinite set {...,-2,-1}
Positive Integer	Number (Long Integer)
Non Negative Integer	Number (Long Integer)
Non Positive Integer	Number (Long Integer)
Unsigned Int	Number (Long Integer)
Short	(no equivalent)
Unsigned Short	(no equivalent)
Long	(no equivalent)
Unsigned Long	(no equivalent)
Byte	Number (Byte)
Unsigned Byte	Number (Byte)
	Additional constraints

XML-Datentypen und Oracle

XML Data Type	**Oracle**
ID	ROWID
String	If no facets if UNICODE NVARCHAR2 (255) else VARCHAR2 (255) else if UNICODE Size = either length or maxlength If Size <= 2000 NCHAR if Size <= 4000 NVARHCAR2 if Size <= 4 Gigabytes NCLOB else if NON UNICODE Size = either length or maxlength If Size <= 2000 CHAR if Size <= 4000 VARCHAR2 if Size <= 4 Gigabytes CLOB
Normalized String	Refer to String
Token	Refer to String
Name	Refer to String
NCName	Refer to String
anyURi	Refer to String
QName	Refer to String
Notation	Refer to String
Boolean	NUMBER Contraint:Boolean
Float	FLOAT
Double	FLOAT
Decimal	NUMBER
Duration	TIMESTAMP
Datetime	TIMESTAMP
Time	DATE
Date	DATE
gYearMonth	INTERVAL YEAR TO MONTH
gYear	DATE
gMonthDay	DATE

© 2005 Altova GmbH

gDay	DATE
gMonth	DATE
	Bin_hex
	if no facets RAW (255)
	Size = either length or maxlength
	If Size <= 2000 RAW (X)
	else Size <= 2 Gigabytes LONG RAW (X)
	if Size <= 4 Gigabytes BLOB (X)
Bin_base64	BLOB
Integer	NUMBER
Int	NUMBER
Negative Integer	NUMBER
	Constraint: restricted to the infinite set {...,-2,-1}
Positive Integer	NUMBER
	Constraint: restricted to the infinite set {...,-2,-1}
Non Negative Integer	NUMBER
	Constraint: restricted to the infinite set {0, 1, 2, ...}
Non Positive Integer	NUMBER
	Constraint: restricted to the infinite set {...,-2 ,-1 ,0}
Unsigned Int	NUMBER
	Additional constraints
Short	NUMBER
Unsigned Short	NUMBER
	Additional constraints
Long	NUMBER
Unsigned Long	NUMBER
	Additional constraints
Byte	BLOB
Unsigned Byte	BLOB
	Additional constraints

XML-Datentypen und MySQL

XML Data Type	**My SQL**
ID	varchar(255)
String	If no facets varchar (255) else if facet length is set and <= 255 = char else if facet maxlength is set and <= 255 = varchar else if maxlength is set and <= 65545 = text else if maxlength is set and <= 16777215 = mediumtext else if maxlength is set and <= 429496295 = longtext
Normalized String	Refer to String
Token	Refer to String
Name	Refer to String
NCName	Refer to String
anyURi	Refer to String
QName	Refer to String
Notation	Refer to String
Boolean	tinyint(1)
Float	float
Double	double

© 2005 Altova GmbH

Decimal	decimal
Duration	timestamp
Datetime	datetime
Time	time
Date	date
gYearMonth	timestamp(4)
gYear	year(4)
gMonthDay	timestamp(8)
	additional constraints to check the month and day value
gDay	timestamp(8)
	additional constraints to check the day value
gMonth	timestamp(8)
	additional constraints to check month value
Bin_hex	If no facets blob (255)
	else if facet length is set and <= 255 = blob
	else if facet maxlength is set and <= 255 = tinyblob else if maxlength is set and <= 65545 = blob
	else if maxlength is set and <= 16777215 = mediumblob else if maxlength is set and <= 429496295 = longblob

Bin_base64	Refer to Bin_hex
Integer	Integer
Int	int
Negative Integer	Integer
	Constraint: restricted to the infinite set {...,-2,-1}
Positive Integer	Integer
	Constraint: restricted to the infinite set {1, 2, ...}
Non Negative Integer	Integer
	Constraint: restricted to the infinite set {0, 1, 2, ...}
Non Positive Integer	Integer
	Constraint: restricted to the infinite set {...,-2 ,-1 ,0}
Unsigned Int	Int
	Additional constraints
Short	Smallint
Unsigned Short	Smallint
	Additional constraints
Long	Bigint
Unsigned Long	Bigint
	Additional constraints
Byte	Tinyint
Unsigned Byte	Tinyint
	Additional constraints

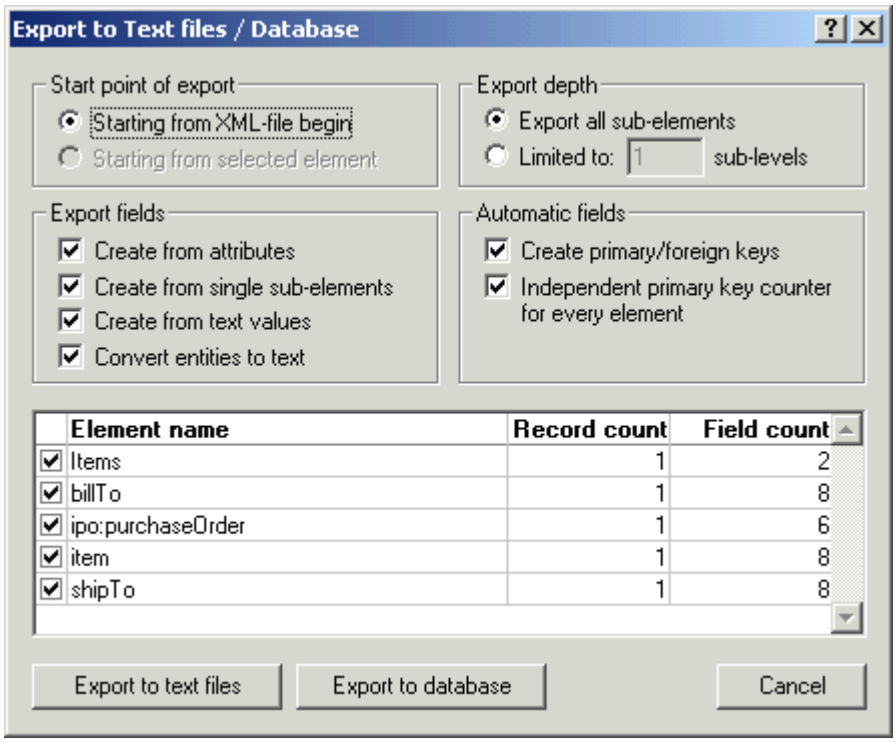
13.9.7 Export in Textdatei/Datenbank

Mit diesem Befehl **exportieren** Sie **XML-Daten** in andere Formate zum Austausch mit Datenbanken oder älteren Systemen.

Abhängig vom Format der Datenausgabe, können Sie entweder XSLT Transformationen oder diesen Exportbefehl zum Exportieren von Daten verwenden.

Zuerst müssen Sie die Struktur der zu exportierenden Daten definieren. Da XML hierarchisch aufgebaut ist, und die meisten Datenbanken und ältere Systeme mit einem relationalen Modell arbeiten, hilft Ihnen XMLSpy 2005 beim Erstellen von Ausgabeformaten, die in relationalen Kontexten interpretiert werden können.

© 2005 Altova GmbH



Export beginnt
Sie können entweder das komplette XML-Dokument exportieren, oder mit dem Export bei einem ausgewählten Element (und dessen Child-Elementen) beginnen.

Exportfelder erzeugen aus:
Abhängig von Ihren XML-Daten, können Sie Elemente, Attribute oder Textinhalte in die Felder (ähnlich wie bei Textdateien oder Datenbanken) exportieren.

Mit der Liste der verfügbaren Elementtypen, die im Vorschau-Fenster angezeigt werden, können Sie entscheiden, welche Elemente exportiert werden sollen. Weiters wird angezeigt, wie viele Felder und Datensätze durch den Export erzeugt werden.

Exporttiefe
Sie können alle Unterelemente exportieren, oder definieren, wie viele Unterelement-Ebenen exportiert werden sollen.

Generiere automatische Felder:
XMLSpy 2005 erstellt für jeden markierten Elementtyp je eine Ausgabedatei oder eine Tabelle. Sie können entweder automatisch Primär-/Sekundärschlüssel-Paare erstellen lassen, um Ihre Daten im relationalen Modell zu verknüpfen, oder für jedes einzelne Element einen Primärschlüssel definieren.

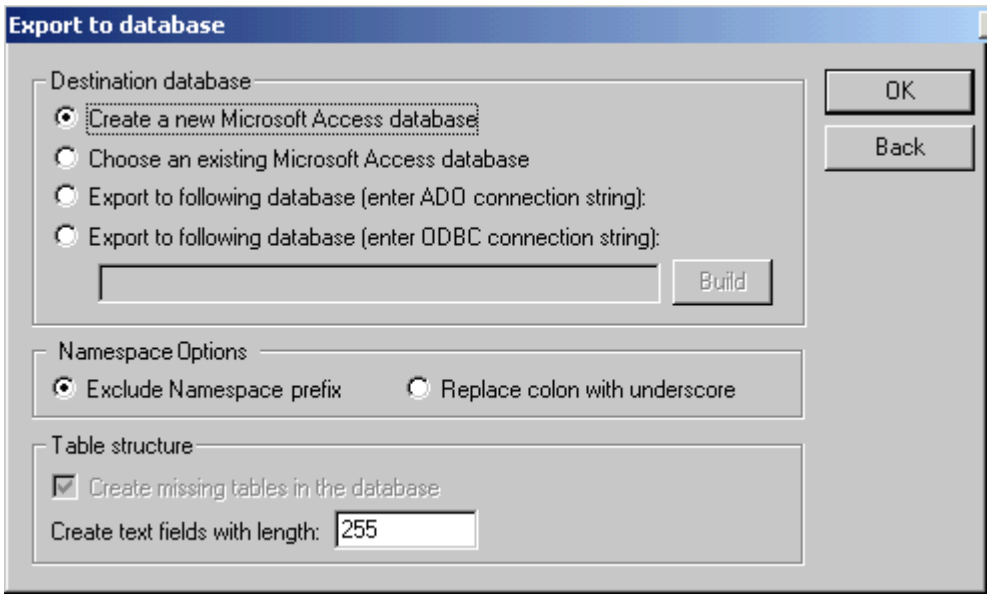
Wenn Sie den Rahmen für die Export-Operation erstellt haben, klicken Sie auf eine der beiden "Export" Schaltflächen, je nachdem, ob Sie Ihre Daten in eine Textdatei oder eine Datenbank exportieren möchten.

Export in Datenbank

Mit diesem Befehl spezifizieren Sie, in welche Art von Datenbank Ihre Daten exportiert werden sollen.

Wählen Sie die Zieldatenbank und die Aktion, die ausgeführt werden soll:

- Erstellen Sie eine neue Microsoft Access (Jet Engine) Datenbankdatei
- Fügen Sie Ihre Daten in Tabellen einer bereits existierenden Access Datei ein, oder
- Exportieren Sie Ihre Daten in ein beliebiges ODBC oder ADO-kompatibles Datenbanksystem (wie SQL Server oder Oracle).
- Die Namespace Optionen erlauben es, die Namespace Präfixe nicht mit einzuschließen, oder den Namespace Präfix Doppelpunkt gegen Unterstriche (_) auszutauschen.

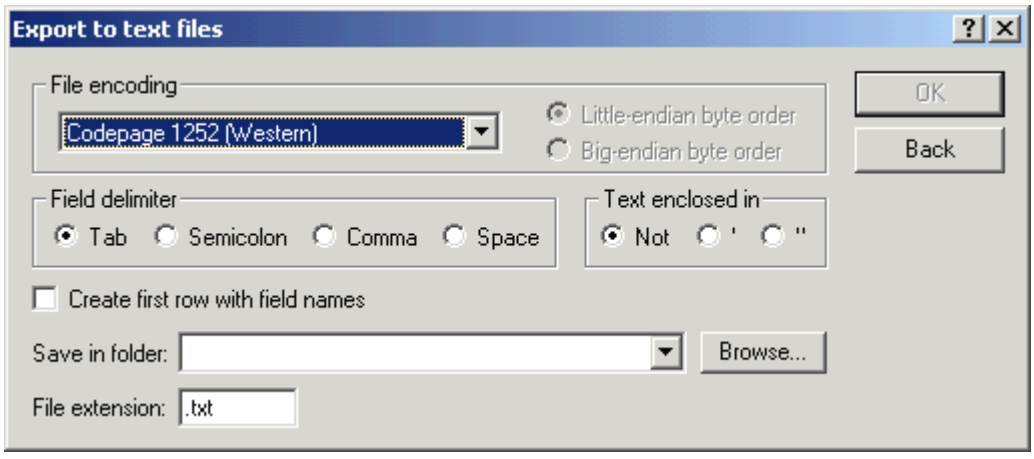


Mit der Schaltfläche "Erzeugen" können Sie Ihren eigenen ADO Connection String erzeugen. Weitere Informationen zu diesem Thema finden Sie unter Datenbank importieren und ADO Connection String.

Export in Textdatei

Mit diesem Befehl können Sie die Formatierung der Textdateien, die exportiert werden sollen, ändern.

Wenn Sie XML-Daten in Textdateien exportieren, müssen Sie die gewünschte Zeichensatzkodierung angeben. Diese Option ist auch im Dialogfeld Textdatei importieren verfügbar.



 © 2005 Altova GmbH

Der Textdateisatz wird in dem Ordner erzeugt, den Sie zuvor angegeben haben. Jeder Dateiname wird aus dem zugehörigen Elementnamen erstellt. Weiters können Sie die gewünschte Dateierweiterung angeben, die verwendet werden soll.

13.9.8 Tamino

Software AG und Altova Inc. haben ein weltweit gültiges Abkommen abgeschlossen, mit dem User von XMLSpy 2005 einfachen und günstigen Zugang zu Tamino erhalten. Mit dieser Partnerschaft erreicht Software AG die ständig wachsende Gemeinschaft von Developern, die Produkte von Altova verwenden.

Dank dieses Abkommens erhalten User von Altova's XMLSpy 2005 und Software AG's Tamino ein integriertes Produktbündel. Tamino XML Server steht auf multiplen Plattformen von PC bis Mainframe zur Verfügung.

XMLSpy 2005 und Tamino (Limited Edition) erhalten Sie als Download und zum Erwerb über unsere Bestellseite im Web. Die Tamino Datenbankkapazität liegt je nach Anzahl der erworbenen Lizenzen für XMLSpy 2005 zwischen 50MB (für XMLSpy 2005 Enterprise/Professional Einzellizenz) bis 1000 MB.

Nach erfolgtem Zahlungseingang erhalten Sie eine Email mit Angaben zum Download von Tamino.

Installation von Tamino

Nachdem Sie das Tamino und XMLSpy 2005 Bündel heruntergeladen haben, können Sie mit der Installation beginnen.

Allgemeine Installationsschritte:

- Download und Installation von XMLSpy 2005
- Download von Tamino XML Server
- Installation von Apache Server (falls noch nicht installiert)
- Installation von Tamino Server 3.1.1
- Erzeugen einer Datenbank in Tamino
- XMLSpy 2005 verwenden

Spezifische Tamino-Installationsanweisungen liegen im HTML-Format vor. Öffnen Sie hierzu die Tamino Dokumentation unter:

<Root>/Windows/INO/Docu/overview.htm

"<**Root**>" steht für das Verzeichnis, in welches Sie Tamino entpackt haben. Wählen Sie den Eintrag "Installing Tamino" (Tamino installieren).

Umfang der vorliegenden Dokumentation:
Es wird davon ausgegangen, dass die Installation von XMLSpy 2005 Enterprise oder Professional Edition, sowie Tamino Server 3.1.1. erfolgreich durchgeführt wurde.

Installation von WebDAV

Installation und Verwendung von WebDAV:
XMLSpy 2005 unterstützt WebDAV. Die Installation von WebDAV ist optional. Die Tamino Funktionalitäten für XMLSpy 2005 sind aufgrund von Synchronisationseinschränkungen nicht mit WebDAV kompatibel.

Installation von WebDAV:

- Installieren Sie WebDAV Server von Software AG.
- Erstellen Sie einen WebDAV Speicher, und starten Sie den WebDAV Server erneut.

Tamino Schema-Extensions

Während XML Schema alle logischen Aspekte der Dokumenttyp-Definition (DTD) berücksichtigt, wird die Definition physischer Aspekte nicht behandelt. Viele XML Prozessoren benötigen deshalb zusätzliche Informationen beim Verarbeiten bestimmter Klassen von Dokumenten.

Tamino benötigt beispielsweise eine Fülle weiterer Informationen über die Dokumentsammlung (Datensammlung), der ein bestimmtes Dokument angehört, welche Elemente für die Indizierung verwendet werden sollen, welche Indizierung verwendet werden soll, wie Elemente eines Dokuments auf Felder in externen Datenbanken gemappt werden sollen, usw...

Hierfür bietet XML Schema einen Extensionsmechanismus. Jedes Schema, Element oder Attribut kann mit einer oder mehreren Annotationen versehen werden.

Jede **Annotation** kann aus zwei Child Elementen bestehen: **documentation** und **appinfo**. Das Element documentation enthält lesbare Dokumentation, während appinfo Informationen zur maschinellen Verarbeitung enthält.

Tamino verwendet **appinfo** zum Speichern von Tamino-bezogenen Informationen in einer Schemadatei:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema"
        xmlns:tsd =
   "http://namespaces.softwareag.com/tamino/TaminoSchemaDefinition">
  <xs:annotation>
    <xs:appinfo>
      <tsd:schemaInfo name = "album">
        <tsd:collection name = "encyclopedia"/>
        <tsd:doctype name = "album">
          <tsd:logical>
            <tsd:content>open</tsd:content>
            <tsd:accessOptions>
              <tsd:read/>
              <tsd:insert/>
              <tsd:delete/>
              <tsd:update/>
            </tsd:accessOptions>
          </tsd:logical>
        </tsd:doctype>xsd:schemaInfo>
    </xs:appinfo>
  </xs:annotation>
...
</xs:schema>
```

Diese Annotationen werden vom Tamino Schema Editor erzeugt, sowie in der XMLSpy 2005 Schema Design Ansicht, wenn Sie mit Tamino Erweiterungen arbeiten.

Erstellen, Speichern und Auflisten von Tamino Schemas

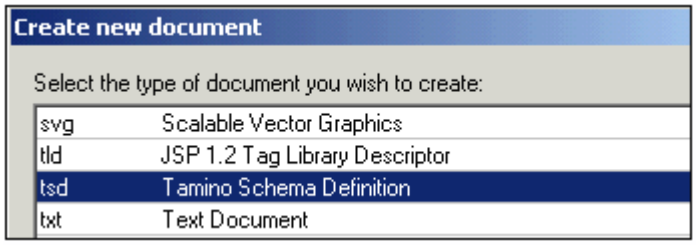
Jede Kommunikation zwischen Tamino und XMLSpy 2005 erfolgt über das HTTP Kommunikationsprotokoll.

 © 2005 Altova GmbH

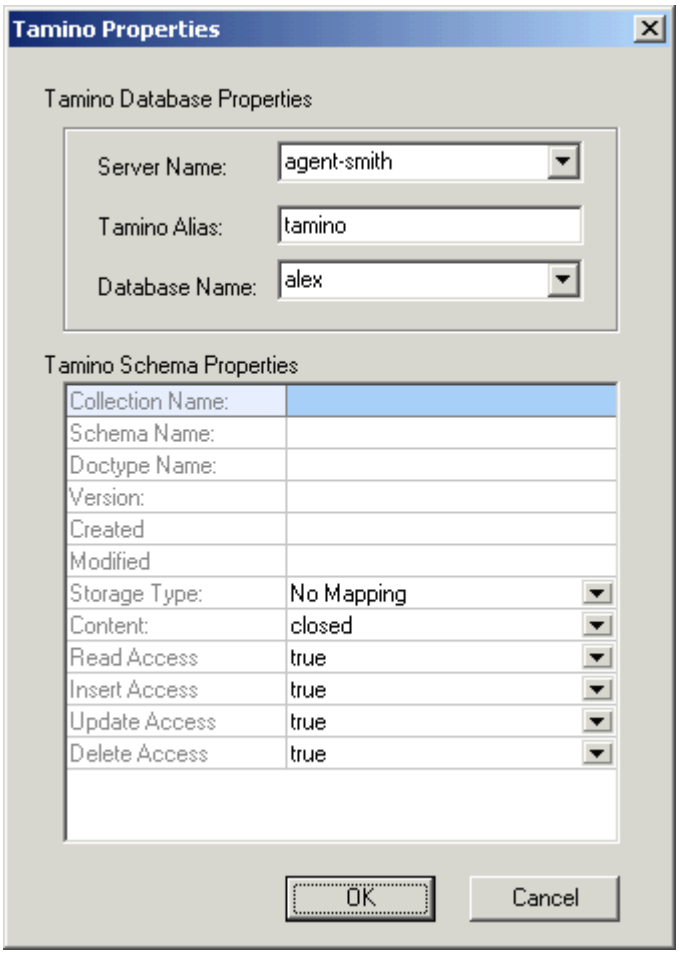
Tamino Datenbank befüllen
Um Daten in Tamino speichern zu können, ist das Erstellen und Speichern eines Schemas in Tamino nötig. Im Weiteren wird ein einfaches Schema erzeugt. Dieses Schema finden Sie auch im Verzeichnis Examples unter dem Namen **BMW.tsd**.

So erstellen Sie eine Tamino Schema Definitionsdatei:

1. Erstellen Sie ein Schema über die Menüoption **Datei | Neu** in XMLSpy 2005, und wählen Sie den Eintrag **tsd Tamino Schema Definition**.



Das Dialogfeld Tamino Eigenschaften wird geöffnet. Hier definieren Sie die Eigenschaften der Datenbank.

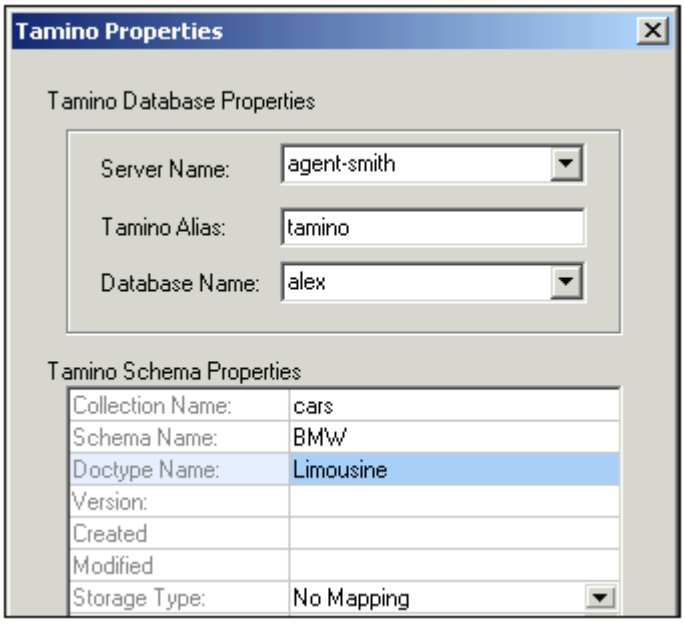


Falls noch nicht angegeben, geben Sie die Namen für Server, Datenbank und Tamino Alias in die entsprechenden Felder ein.

2. Geben Sie den Namen der Datensammlung, des Schemas und des Doctypes in die entsprechenden Felder ein, und bestätigen Sie mit OK. Die hier angegebenen Informationen werden in Tamino übernommen.

Datensammlung Name:	cars
Schema Name:	BMW
Doctype Name:	Limousine

Bitte beachten Sie:
Der Tamino Alias wird vom Datenbank-Administrator aus Sicherheitszwecken vergeben, oder wenn ein Alias bereits besetzt ist. Bitte wenden Sie sich an Ihren Datenbank-Administrator für weitere Informationen zu Tamino Alias.



Eine Tamino Schema Definitionsdatei wird erzeugt, die in der Schema Design Ansicht dargestellt wird.

3. Geben Sie den Doctype Name "Limousine" im Root Element-Feld ein. (Der Doctype Name kann ein beliebiges globales Element sein, und muss nicht unbedingt das Root Element sein. Sie benötigen diesen Namen allerdings in der Schema-Datei zum Speichern in Tamino.)

4. Jetzt ist die Schemadatei mit der Tamino Datenbank kompatibel. In der Eingabehilfe Details wurde ein Register mit Namen "Tamino" angehängt. Tamino Extensionen können über die Eingabehilfe Details für Elemente und Attribute zugewiesen werden. Durch Klicken auf das Register Text-Ansicht wird die Schemadefinition wie folgt dargestellt:

 © 2005 Altova GmbH

```
<xs:schema
xmlns:tsd="http://namespaces.softwareag.com/tamino/TaminoSchemaDefiniti
on" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:appinfo>
            <tsd:schemaInfo name="BMW">
                <tsd:collection name="cars"/>
                <tsd:doctype name="Limousine">
                    <tsd:logical>
                        <tsd:content>closed</tsd:content>
                    </tsd:logical>
                </tsd:doctype>
            </tsd:schemaInfo>
        </xs:appinfo>
    </xs:annotation>
    <xs:element name="Limousine">
        <xs:annotation>
            <xs:documentation>The root element is usually named the same
as the doctype.</xs:documentation>
        </xs:annotation>
    </xs:element>
</xs:schema>
```

5. Entwerfen Sie Ihr Schema mit dem Schemaeditor wie folgt:

   Hängen Sie am Root-Element "Limousine" einen Sequenz-Node an, und danach folgende Child Element Nodes:

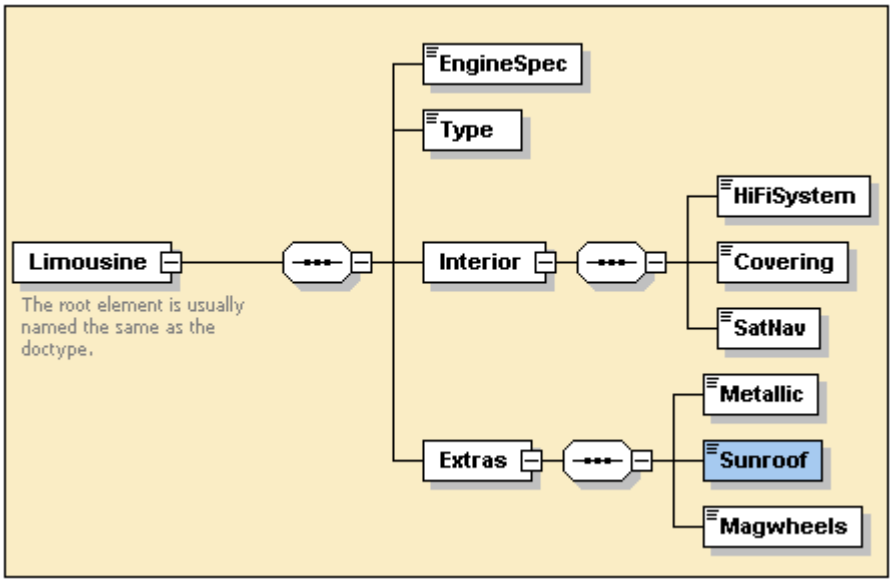
EngineSpec	xs:string
Type	xs:string
Interior	
Extras	

   Hängen Sie einen Sequenz-Node an das Element "Interior" an, und dann folgende Children:

HiFiSystem	xs:string
Covering	xs:string
SatNav	xs:string

   Hängen Sie einen Sequenz-Node an das Element "Extras" an, und dann folgende Children:

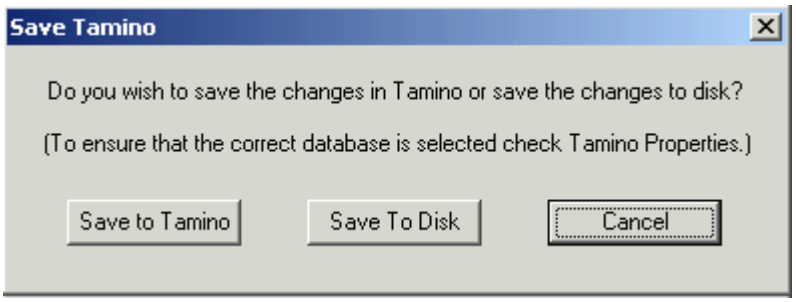
Metallic	xs:boolean
Sunroof	xs:boolean
Magwheels	xs:boolean



Bitte beachten Sie:
Tamino unterstützt keine **Global complex** types!

So speichern Sie ein Tamino Schema:

1. Verwenden Sie die Tastenkombination Strg+S, und klicken Sie dann auf die Schaltfläche "In Tamino speichern".
   Das Schema wird in der Tamino Datenbank gespeichert. Sie können auch die Menüoption **Konvertieren | Tamino | In Tamino speichern...** wählen, um das Schema direkt in der Datenbank zu speichern.



Bitte beachten Sie:
Wenn Sie ein Schema in Tamino speichern, wird automatisch der **Tamino Schema Validator** aufgerufen. Dieser Validator überprüft Tamino-bezogene Informationen wie Extensionen und spezifische Regeln, und ist nicht mit dem Validator in XMLSpy 2005 zu verwechseln. Bei Tamino Validierungsfehlern erhalten Sie eine entsprechende Fehlermeldung. Weitere Informationen zu diesem Thema erhalten Sie in der Tamino Dokumentation.

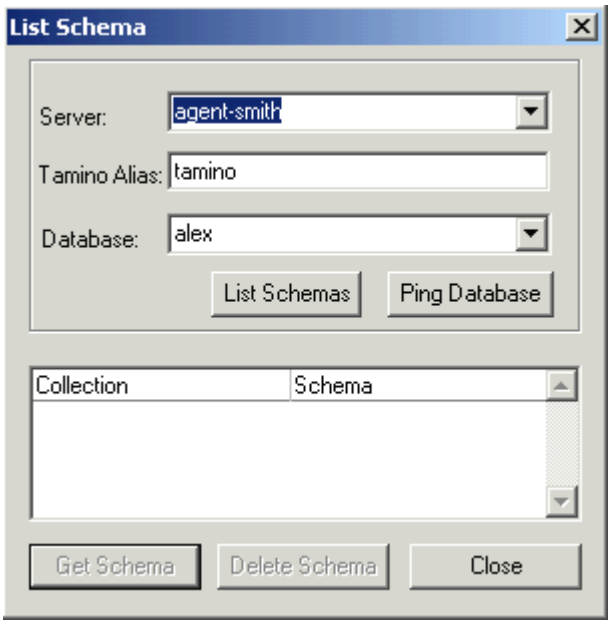
Auf Festplatte speichern
Mit der oben genannten Methode wird das Dokument in der Datenbank gespeichert, ohne es auf der Festplatte zu speichern. Im Hauptfenster wird als Name des Dokuments **Untitledxx.xsd** angezeigt. Um die Datei lokal zu speichern, verwenden Sie die Schaltfläche **"Auf Festplatte speichern"**.

Wenn Sie XML Dateien in XMLSpy 2005 erzeugen möchten, die auf Schemas basieren, die in Tamino gespeichert wurden, verwenden Sie immer den Befehl "Auf Festplatte speichern", sodass eine Kopie des Schemas lokal vorhanden ist.

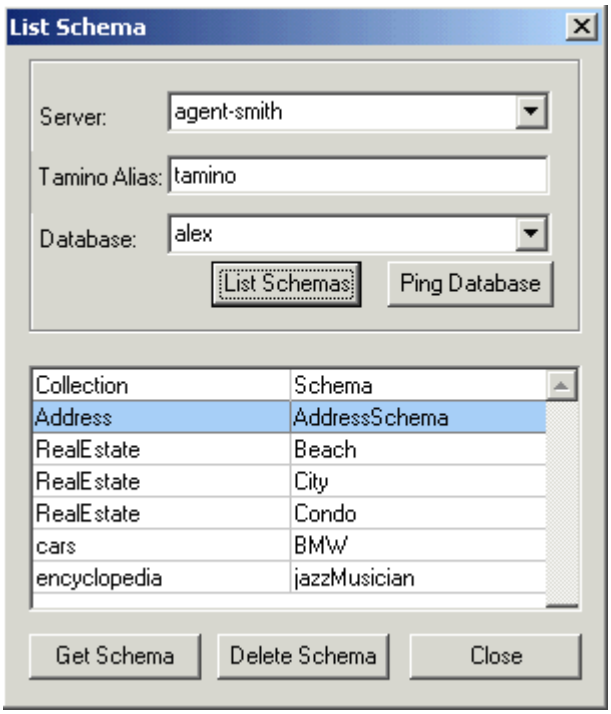
 © 2005 Altova GmbH

So können Sie Schemas in der Tamino Datenbank auflisten:

1. Wählen Sie die Menüoption **Konvertieren | Tamino | Schema auflisten...**
   Das Dialogfeld "Schema auflisten" wird geöffnet.



2. Falls noch nicht angegeben, geben Sie Server- und Datenbanknamen in den entsprechenden Feldern ein.
3. Klicken Sie auf die Schaltfläche **Auflisten**.
   Eine Liste mit Schemas und verknüpften Datensammlungen der aktuellen Datenbank wird angezeigt.



Anmerkung: Über die Schaltfläche "Ping Database" können Sie die Verbindung zu Tamino überprüfen.

So öffnen Sie ein Schema:

- Doppelklicken Sie auf den Schema-Namen im Listenfeld, oder
- Markieren Sie den entsprechenden Eintrag, und klicken Sie auf "Schema holen".

So löschen Sie ein Schema aus Tamino:

1. Zeigen Sie die Schemas im Dialogfeld "Schema auflisten".
2. Markieren Sie das zu löschende Schema, und klicken Sie auf "Schema löschen".

Validierung von Tamino Schemas

- Nachdem Sie ein Schema aus Tamino "geholt" haben, wählen Sie die Menüoption **XML | Validieren**, oder betätigen Sie die Funktionstaste F8.
  In diesem Fall wird der Validator von XMLSpy 2005 verwendet. XMLSpy 2005 Fehlermeldungen werden bei ungültigen Schemas angezeigt.

Wenn Sie ein Schema **in Tamino speichern**, wird automatisch der **Tamino Schema Validator** aufgerufen. Dieser Validator überprüft Tamino-bezogene Informationen wie Extensionen und spezifische Regeln, und ist nicht mit dem Validator in XMLSpy 2005 zu verwechseln. Bei Tamino Validierungsfehlern erhalten Sie eine entsprechende Fehlermeldung. Weitere Informationen zu diesem Thema erhalten Sie in der Tamino Dokumentation.

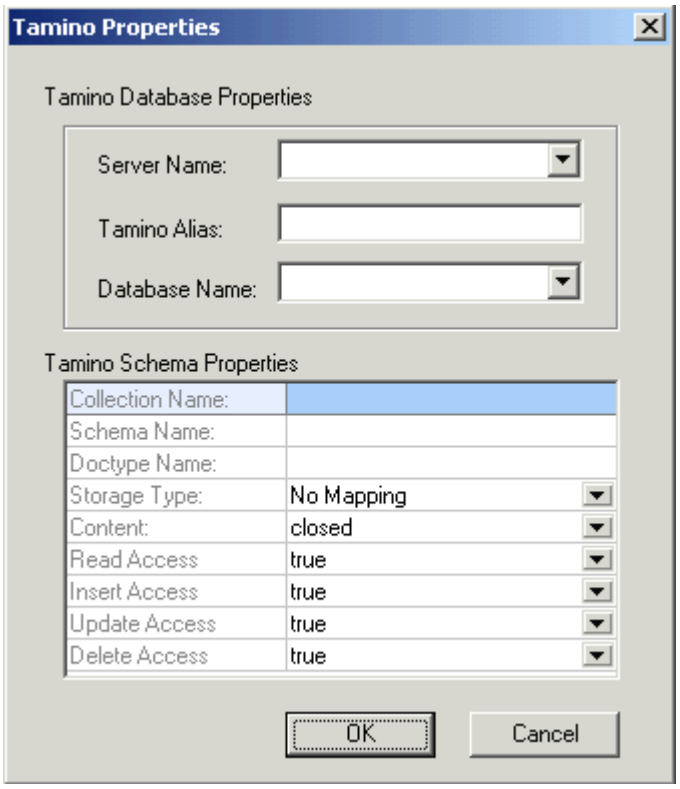
Legacy Schemas in Tamino hinzufügen

Schemas, die in XMLSpy 2005 erzeugt wurden, können auf einfache Weise in Tamino Server hinzugefügt werden.

So fügen Sie ein existierendes Schema in Tamino Server hinzu:

1. Öffnen Sie das Schema in der Schema/WSDL Design-Ansicht.
2. Wählen Sie die Menüoption **Schema-Design | Tamino Schema-Extensions aktivieren**.
   Wenn Sie ein anderes Präfix als 'xs' verwenden, wird ein Dialogfeld angezeigt. In Tamino muss das Präfix immer **'xs'** sein.
3. Klicken Sie auf OK, falls das Dialogfeld angezeigt wird.
   Ein weiteres Dialogfeld wird angezeigt, und Sie werden gefragt, ob die Dateierweiterung in .tsd geändert werden soll. Dies ist optional - zum Bestätigen klicken Sie bitte auf OK.
4. Das Dialogfeld Tamino Eigenschaften wird angezeigt. Geben Sie die Eigenschaften der Datenbank sowie die des Schemas an: Datensammlung, Schema und Doctype, und bestätigen Sie mit OK.

© 2005 Altova GmbH



5. Geben Sie den Namen für Datensammlung (Collection), Schema und Doctype ein.
6. Wählen Sie die Menüoption **Konvertieren | Tamino | In Tamino speichern**.
   Sie erhaltern eine Bestätigungsmeldung, sobald das Schema erfolgreich gespeichert wurde.

Einschränkungen für Tamino Schema Definitionen

Einschränkungen der Tamino Schema Definition Language-Implementierung (Copyright Software AG 2001)
Die in der aktuellen Version von Tamino implementierte Tamino Schema Definition Language weist die folgenden Einschränkungen auf:

Logische TSD3-Einschränkungen, die nicht durch eine Syntax-Untermenge eines logischen W3C XS Meta-Schemas abgeleitet werden können:

- Die Element-Namen jedes einzelnen lokalen Elements und aller referenzierten globalen Elemente müssen für jede Model Group (d.h. den Inhalt von 1 complexType) eindeutig sein. Das bedeutet, dass für jeden Element-Namen entweder ein lokales Element oder mehrere Referenzen auf globale Elemente zulässig sind.
- Für das Attribut xs:type und xs:base sind nur in XML-Schema, Teil 2, definierte Datentypen zulässig; siehe auch kommentierte Definition von tsd:native-Datatype in dieser Dokumentation.
  Logische TSD3 Einschränkungen im Vergleich zu vollständigen DTD-Funktionalitäten, die in einem vollständigen XML-Schema ausgedrückt werden können (z.B. allgemeine Entitäten).
- Enumerationen für Elemente ohne Unterelemente und mit Attributen können nicht definiert werden (Einschränkung für XML-Schema: Dieses kann nicht ohne named complex types ausgedrückt werden.)
  Physische TSD3 Einschränkungen
- Anhängen einer physischen Schema-Definition ist nicht möglich ohne das Element tsd:which unterhalb von rekursiven Elementen.

- Für jedes Schema, Element oder jede attribute-definition ist nur ein schemaInfo-, elementInfo-, attributeInfo-Element möglich.

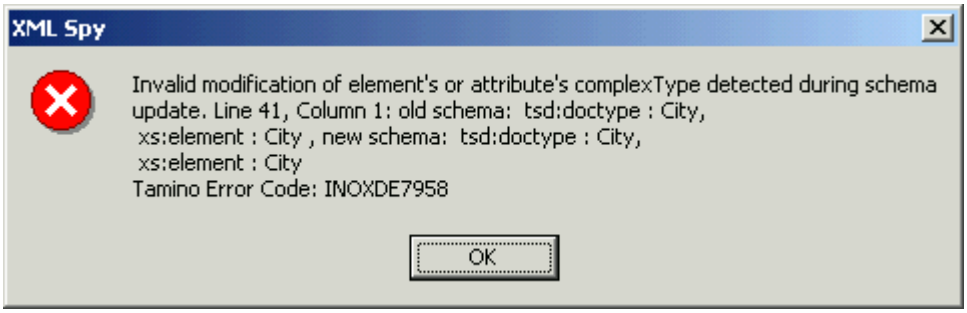
Einschränkungen von XMLSpy 2005 und dem Tamino Paket

- Transaktionsorientierte Verarbeitung wird nicht unterstützt.
- Nur XML-Dokumente und XSD-Schema-Dokumente können über XMLSpy 2005 in Tamino gespeichert werden.
- Inkludierung von Schemas innerhalb von Schemas wird nicht unterstützt.
- Import von Schemas wird nicht unterstützt.
- Namespaces (Präfixe) in einem Schema werden von Tamino Server nicht unterstützt.
- Die in XMLSpy 2005 verfügbaren Tamino-Funktionen sind nicht mit WebDAV kompatibel.

Bearbeiten von Tamino Schemas

Bearbeiten von Tamino Schemas (die XML Dokumente in der Datenbank referenzieren)
Wenn Sie eine Schema-Definition ändern, auf der mehrere XML-Dokumente basieren, kann dies zur Folge haben, dass die verknüpften XML-Dokumente nicht mehr gültig sind. Beim Speichern prüft Tamino, ob die Schemas noch gültig sind. Falls nicht, erhalten Sie eine Tamino-Fehlermeldung.

Beispiel:
Ein Schema und ein verknüpftes Dokument sind in Tamino vorhanden. Das Schema wird in XMLSpy 2005 geöffnet, und ein neues obligatorisches Element wird an einen Complex Type angehängt. Wenn Sie dieses Schema nun in Tamino speichern, erhalten Sie folgende Fehlermeldung.



Wenn Sie nun das Element von obligatorisch in optional ändern (min=0 and max=1), kann das Schema in Tamino gespeichert werden, ohne dass die referenzierten XML-Dokumente ungültig werden.

Erstellen von XML-Dateien, die auf Tamino Schemas basieren

So erstellen Sie XML-Dateien, die auf Tamino basieren:
Sie können XML-Dateien, die auf Schemas basieren, auf zwei verschiedene Arten erstellen:

- Verwendung eines bereits existierenden XML-Dokuments als **Vorlage**, für die sowohl die XML-, als auch die Schemadatei bereits in Tamino vorhanden ist, oder
- Verwendung einer Schemadatei, die bereits in Tamino existiert, als Basis für Ihr neues XML-Dokument.

So erstellen Sie eine XML-Datei die auf einer anderen XML (Vorlagen-)Datei basiert:

1. Erzeugen Sie eine Abfrage für das XML Dokument, welches als Vorlage verwendet werden soll, und holen Sie das Dokument vom Tamino Server. (Alternativ können Sie auch die Methode Datei holen verwenden.)
2. Bearbeiten Sie das XML-Dokument.
3. Verwenden Sie die Menüoption **Konvertieren | Tamino | Dokument hinzufügen**, um

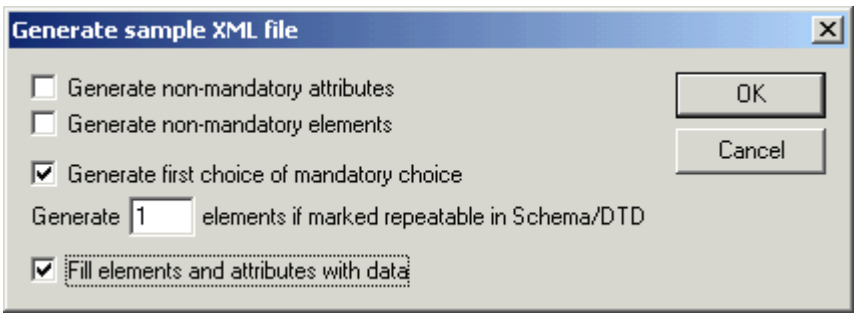
 © 2005 Altova GmbH

das neue XML-Dokument wieder auf dem Tamino Server zu speichern. Beispieldateien finden Sie im Ordner Example, z.B. cars1.xml und cars2.xml.

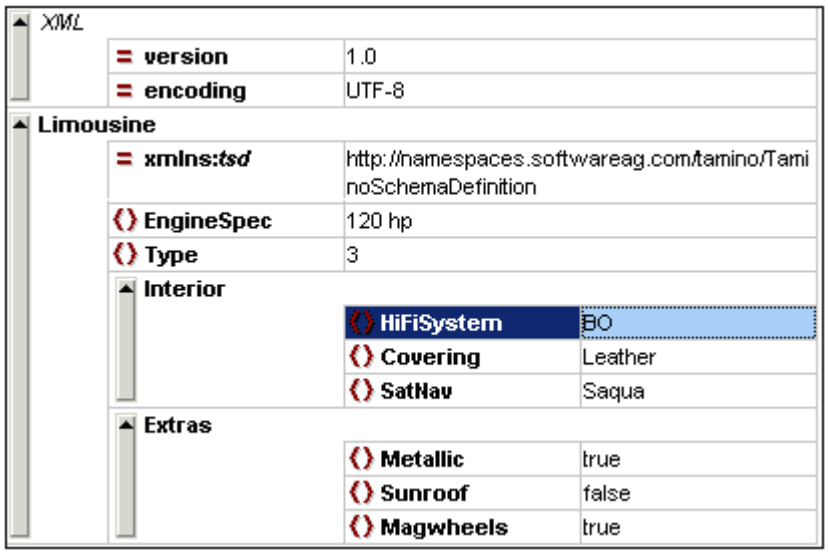
So erstellen Sie eine XML-Datei, die auf einem lokal gespeicherten Tamino Schema basiert

Nachdem Sie die Schemadatei geöffnet haben, die in Tamino gespeichert ist (BMW Schema in diesem Beispiel)

1. Stellen Sie sicher, dass die geöffnete Schemadatei das aktive Dokument ist.
2. Wählen Sie **Konvertieren | Tamino | XML Beispieldatei erzeugen**.



3. Ein Dialogfeld wird angezeigt. Aktivieren Sie die Kontrollkästchen "Erste Wahl von obligatorischer Auswahl generieren" und "Elemente und Attribute mit Daten befüllen". Bestätigen Sie mit OK.
4. Eine XML Beispieldatei wird erzeugt. Ändern Sie nun die Daten wie unten angegeben:

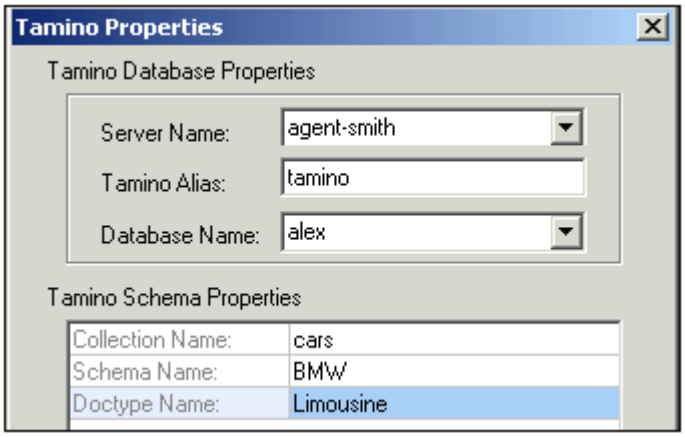


Bitte beachten Sie:

Beim Auswählen der Menüoption **XML | Validieren** wird die XML Datei gegen die **lokale** Tamino Schemadatei validiert (mittels XMLSpy 2005 Validator).

So fügen Sie eine XML Datei, einen Datensatz oder ein Dokument in Tamino ein:

1. Wählen Sie die Menüoption **Konvertieren | Tamino | Dokument hinzufügen** um das aktive XML-Dokument in Tamino hinzuzufügen.
   Ein Dialogfeld wird geöffnet, in dem Sie um weitere Angaben gebeten werden.
2. Geben Sie die Namen der Datensammlung, des Schemas und des Doctype im Fenster Tamino-Eigenschaften ein, und bestätigen Sie mit OK.



Die XML Datei ist nun mit dem in Tamino gespeicherten Schema verknüpft. Sie erhalten eine Bestätigungsmeldung, dass das XML-Dokument erfolgreich hinzugefügt wurde.

3. Ändern Sie einen der Einträge in der offenen XML-Datei in (metallic=false) und wählen Sie nochmals **Konvertieren | Tamino | Dokument hinzufügen**.
   Die bearbeitete XML-Datei wird mit den in Tamino getätigten Einstellungen in die Datenbank aufgenommen. Sie haben nun zwei XML Dateien, die mit der Cars-Datensammlung assoziiert sind, BMW Schema, und Limousine Doctype.

Bitte beachten Sie:

Beim Validieren einer **XML-Datei**, die zum **Tamino Server hinzugefügt** wurde, wird diese gegen die referenzierte Schemadatei in Tamino validiert.

Derzeit können nur XML- und verknüpfte XSD-Dateien über XMLSpy 2005 in Tamino gespeichert werden.

Abfrage von Daten in Tamino

Abfrage der Datenbank in Tamino

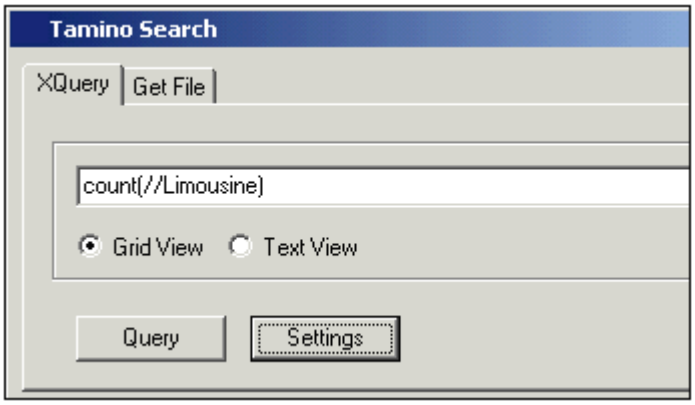
Mit der Menüoption **Konvertieren | Tamino | Suche** definieren Sie eine neue Abfrage. Tamino verwendet die Abfragesprache **XQuery**. Die Syntax dieser Sprache folgt den Vorgaben von Standard XPath Notationen.

Sie können über folgendes Dialogfeld sowohl eine XQuery starten, als auch eine spezifische Datei aus der Datenbank (wenn Sie deren ID kennen) abrufen. Das Resultat der Abfrage wird als XML-Dokument in der Grid- oder Text-Ansicht angezeigt.

Beispiel:
Sie möchten wissen, wie viele Datensätze es mit dem Doctype "Limousine" gibt. Dafür wird der XPath Befehl **count()** zur Abfrage verwendet.

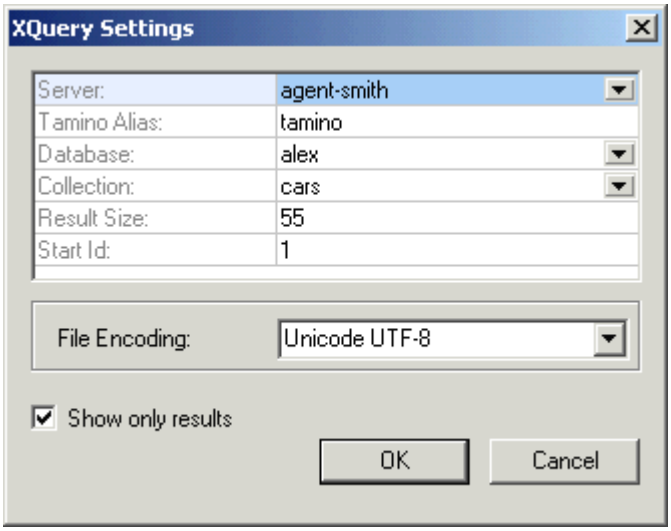
1. Geben Sie **count(//Limousine)** im XQuery Feld ein.

 © 2005 Altova GmbH



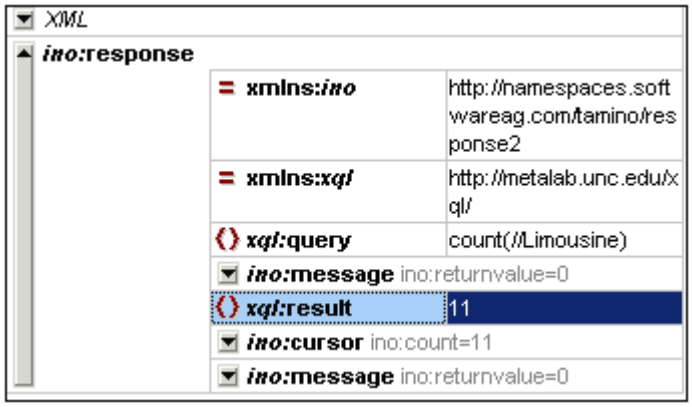
Nachdem das Abfragekriterium bestimmt wurde, müssen Sie noch angeben, welche Datenbank, Datensammlung und Doctype durchsucht werden sollen.

2. Klicken Sie auf die Schaltfläche "Einstellungen", geben Sie die gewünschten Einstellungen an, und bestätigen Sie mit OK. Bitte beachten Sie die **Groß- und Kleinschreibung** bei Ihren Angaben.



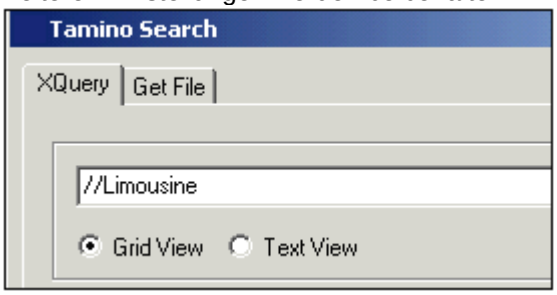
3. Klicken Sie auf die Schaltfläche **Abfrage** und danach Schließen, um das Dialogfeld zu schließen.
   Das Resultat der Abfrage wird als XML-Dokument angezeigt: **xql:result** zeigt in diesem Fall 11 an.

   Bitte beachten Sie, dass die Standardeinstellung für das Feld Result Size: 5 ist. In diesem Beispiel wurde die Zahl auf 55 erhöht. Wenn Sie das Kontrollkästchen "Nur Resultate anzeigen" aktivieren, wird im XML-Dokument nur das Resultat dargestellt.

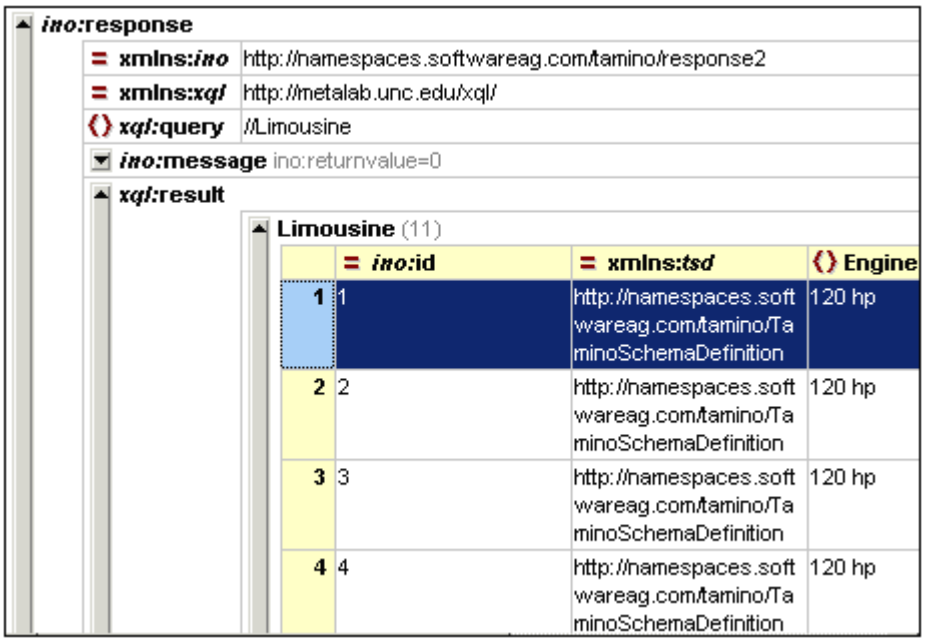


Spezifische Dokumente aus Tamino holen:

1. Definieren Sie eine neue Abfrage, bei der alle XML-Dokumente des Doctypes Limousine angezeigt werden sollen und klicken Sie auf die Schaltfläche Abfrage. Alle weiteren Einstellungen werden beibehalten.

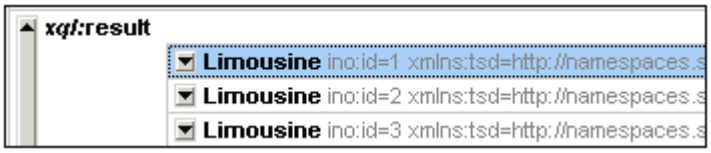


Das Resultat wird angezeigt, in der Tabellen-Ansicht werden mehrere Resultate angezeigt.

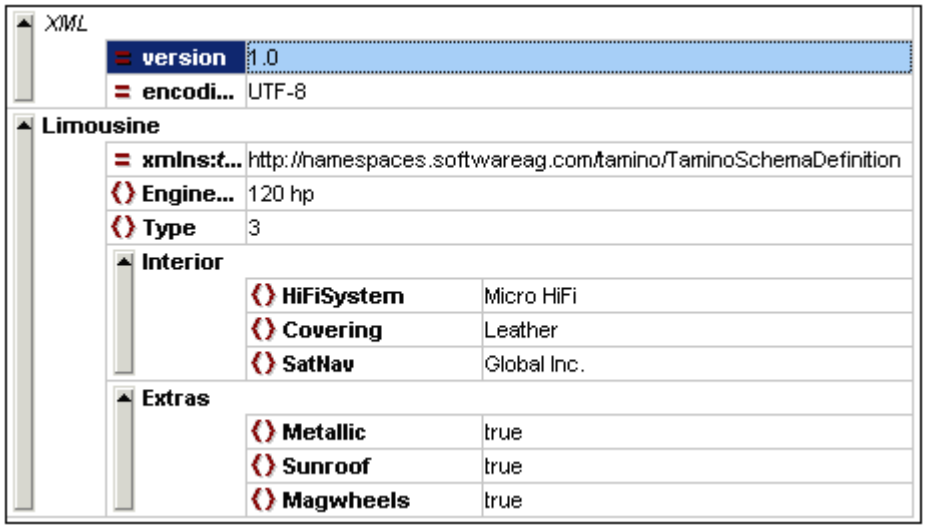


2. Klicken Sie in die Spaltennummer in der Tabelle, und wählen Sie die Menüoption **Konvertieren | Tamino | Aktives Dokument holen**. Wenn die Tabellen-Ansicht nicht aktiviert ist, klicken Sie in die entsprechende Zeile "Limousine".

 © 2005 Altova GmbH

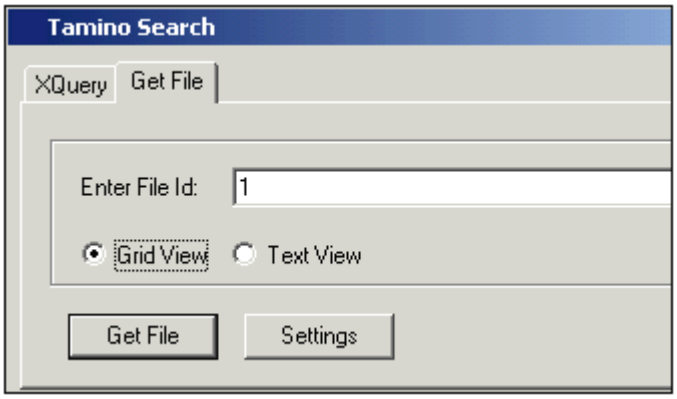


Das Abfrageresultat wird als XML-Dokument im Hauptfenster angezeigt.



Dateimethode zur Suche eines bestimmten Dokuments:

- Notieren Sie die ID Nr. des Abfragedokuments und öffnen Sie erneut das Dialogfeld für die Abfrage. Die Dokument-ID ist der Eintrag **ino:id=x** .
- Klicken Sie auf das Register **Dokument holen**, und geben Sie die ID im Textfeld ID ein.



- Klicken Sie auf **Dokument holen** um das XML-Dokument zu öffnen.

So speichern Sie eine Tamino Abfrage:

Verwenden Sie die Tastenkombination **Strg+S** um die Abfrage lokal zu speichern. Hierdurch wird die Abfrage als XML-Dokument gespeichert. Diese steht allerdings nicht mit dem Tamino Server in Verbindung.

Abfrage aktualisieren:

Nachdem Sie eine Abfrage definiert und ausgeführt haben, kann das Abfragedokument verwendet werden, um diese zu einem späteren Zeitpunkt wieder zu verwenden.

1. Gehen Sie zurück zum ursprünglichen Abfragedokument.

2. Wählen Sie die Menüoption **Konvertieren | Tamino | Aktualisieren**.
   Tamino wird so mit den gleichen Kriterien neu abgefragt. Das neue Abfrageresultat stellt dann ein aktualisiertes Resultat dar.

Bearbeiten von existierenden XML-Dokumenten und Hinzufügen in Tamino
Ändern Sie Daten im XML-Dokument und wählen Sie die Menüoption **Konvertieren | Tamino | Dokument hinzufügen**. Das bearbeitete Dokument wird in Tamino hinzugefügt. Eine Bestätigungsmeldung zeigt an, dass das Dokument erfolgreich hinzugefügt wurde.

13.9.9 Oracle XML DB

Mit XMLSpy 2005 können Sie eine Verbindung zu Oracle XML-Datenbanken herstellen, um diese abzufragen.

Folgende Datenbankfunktionen werden unterstützt:

- Hinzufügen (und Registrieren) von XML-Schemas in einer Oracle XML-Datenbank. Zum Registrieren von XML-Schemas mit XMLSpy 2005 muss zuvor der Oracle XML DB Client installiert werden.
- Öffnen und Löschen von Schemas
- Abfrage der Datenbank durch Verwendung von XPath Statements (DBUri)
- Durchsuchen von XML-Dokumenten (mittels WebDAV)
- Erstellen von XML-Dokumenten basierend auf Schemas, die in der Datenbank gespeichert sind.

Installation:

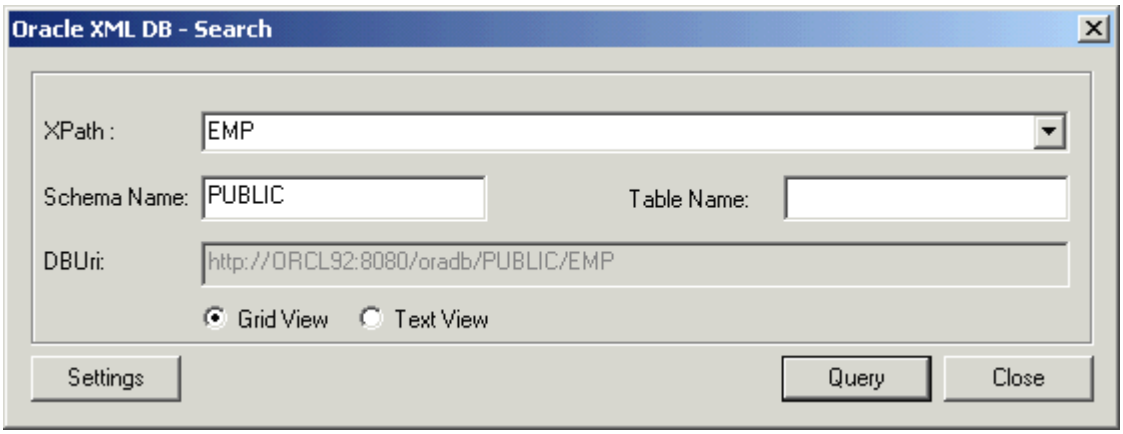
- Downloaden und installieren Sie XMLSpy 2005
- Installieren Sie Oracle Server (falls nötig)
- Erzeugen Sie eine Oracle Datenbank

Unter http://otn.oracle.com/sample_code/tech/xml/xmldb/XDBBasicDemo.pdf finden Sie eine Demoinstallation für eine Oracle XML DB. In diesem Dokument wird beschrieben, wie die Oracle Software installiert und für XMLSpy 2005 konfiguriert wird.

Suchen...

Mit diesem Befehl können Sie die Oracle XML Datenbank durch Verwendung von DBUri Statements abfragen. Geben Sie die Abfrage als XPath im XPath Feld ein. Ihre Abfrage wird im darunter liegenden DBUri Feld angezeigt und wird als http Protokoll an den Server gesendet.

© 2005 Altova GmbH



Das Feld **Schema-Name** muss nur bei der ersten Abfrage eingegeben werden. Bei Folgeabfragen wird der Wert automatisch eingetragen.

Im Feld **Tabellenname** können Sie angeben, welche Tabelle abgefragt werden soll.

Einstellungen
Durch Betätigen dieser Schaltfläche wird das Dialogfeld Einstellungen geöffnet, in dem Sie Abfrageeinstellungen tätigen können: Servername, Port, Benutzername und Passwort.

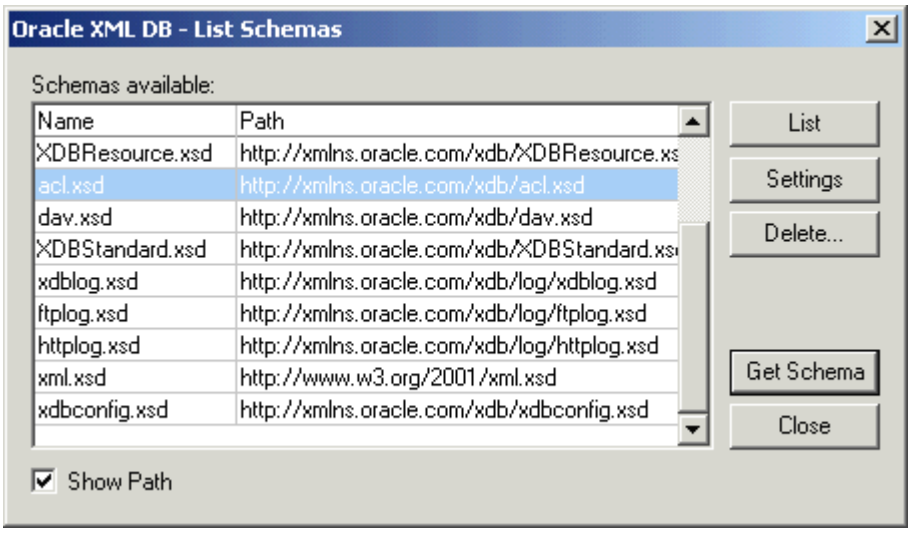
Abfrage
Durch Betätigen dieser Schaltfläche wird die Abfrage abgesendet und das Dialogfeld geschlossen.

Das Resultat der Abfrage erscheint in der Enhanced Grid oder Text-Ansicht, je nachdem, welche Option Sie im Dialogfeld angegeben haben. War die Abfrage nicht erfolgreich, wird eine Fehlermeldung angezeigt.

Schemas auflisten

So zeigen Sie die Liste der verfügbaren Schemas in der Oracle Datenbank an:

1. Wählen Sie die Menüoption **Oracle XML-Datenbank | Schemas auflisten**.
   Das Dialogfeld Oracle XML-Datenbank - Schemas auflisten wird geöffnet.
2. Klicken Sie auf die Schaltfläche **Auflisten** um die Schemas der aktuellen Datenbank anzuzeigen.



3. Doppelklicken Sie auf eines der Schemas in der Liste oder betätigen Sie die Schaltfläche "Schema holen". Das ausgewählte Schema wird in der Schema/WSDL Design-Ansicht angezeigt.

Das Dialogfeld Oracle XML-Datenbank - Schemas auflisten erlaubt es Ihnen, weitere Optionen auszuwählen.

Einstellungen
Mit dieser Schaltfläche definieren oder ändern Sie die Einstellungen der aktuellen Oracle XML Datenbank.

Löschen
Mit dieser Schaltfläche löschen Sie ein eingetragenes Schema (welches im Listenfeld angezeigt wird) aus der Oracle XML Datenbank.
Nachdem Sie die Schaltfläche Löschen betätigt haben,

1. Wählen Sie entweder das Optionsfeld ADO oder ODBC Connection String und klicken Sie auf die Schaltfläche Erzeugen.
   Das Dialogfeld Datenverknüpfungseigenschaften wird geöffnet (oder wählen Sie das Dialogfeld Datenquelle falls Sie ODBC ausgewählt haben).
2. Wählen Sie **MS OLE DB Provider for Oracle** und klicken Sie auf **Weiter**.
3. Geben Sie die Serverinformationen ein: Servername, Username, Passwort, und klicken Sie auf OK. Der ADO Connection String wird nun im Feld Connection String angezeigt.
4. Klicken Sie auf die Schaltfläche Löschen, um das Schema zu entfernen.

Schema holen
Öffnet das ausgewählte Schema in der Schema/WSDL Design-Ansicht.

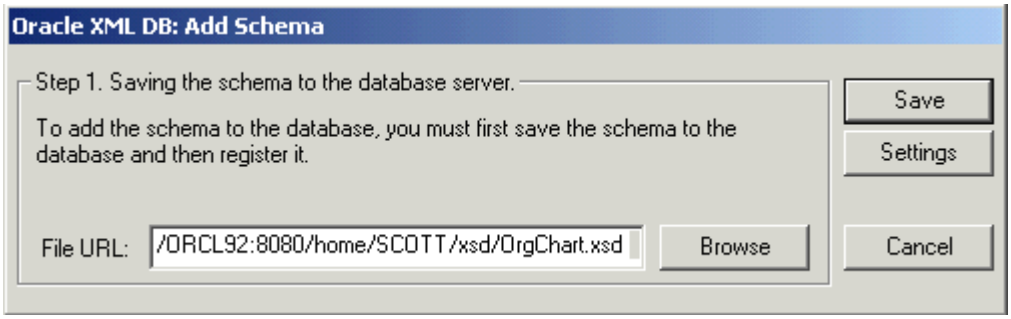
Schließen
Schließt das Dialogfeld.

Schema hinzufügen

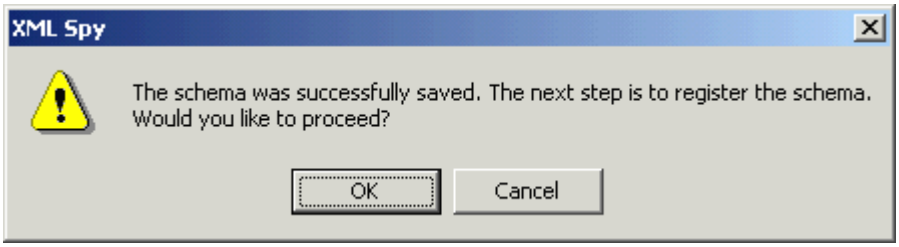
Mit diesem Befehl können Sie ein XML Schema zur Oracle XML Datenbank hinzufügen. Informationen zum Thema XML-Dokument Instanzdatei in die Datenbank einfügen, finden Sie unter XML-Dokumentinstanzen hinzufügen.

© 2005 Altova GmbH

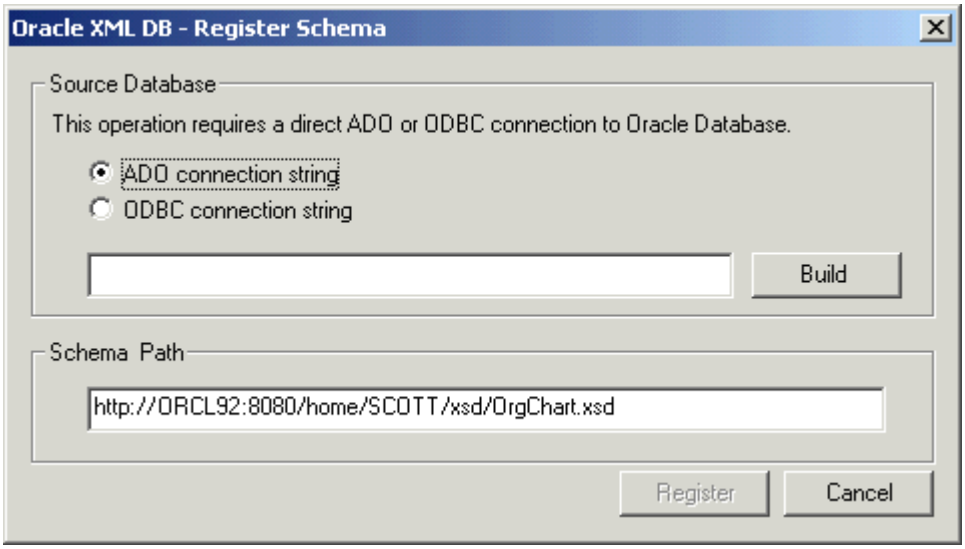
1. Wählen Sie die Menüoption **Oracle XML-Datenbank | Schema hinzufügen**.
   Dies ist der erste Schritt beim Hinzufügen eines Schemas. Das Schema wird hierdurch in der Datenbank gespeichert. Die URL der Datei wird automatisch im Textfeld Datei URL angezeigt.
   Klicken Sie auf die Durchsuchen-Schaltfläche um anzugeben, wo die Datei gespeichert werden soll. Geben Sie den Dateinamen der neuen Schemadatei ein.



2. Klicken Sie auf **Speichern** um das Schema in der Datenbank zu speichern. Danach erhalten Sie eine Meldung, dass das Schema erfolgreich in der Datenbank gespeichert wurde. Mit OK registrieren Sie nun das Schema.



   Das Dialogfeld **Schema registrieren** wird geöffnet. Der Pfad des soeben gespeicherten Schemas wird im Textfeld Schemapfad angezeigt.
3. Klicken Sie auf das Optionsfeld ADO oder ODBC Connection String und dann die Schaltfläche Erzeugen.



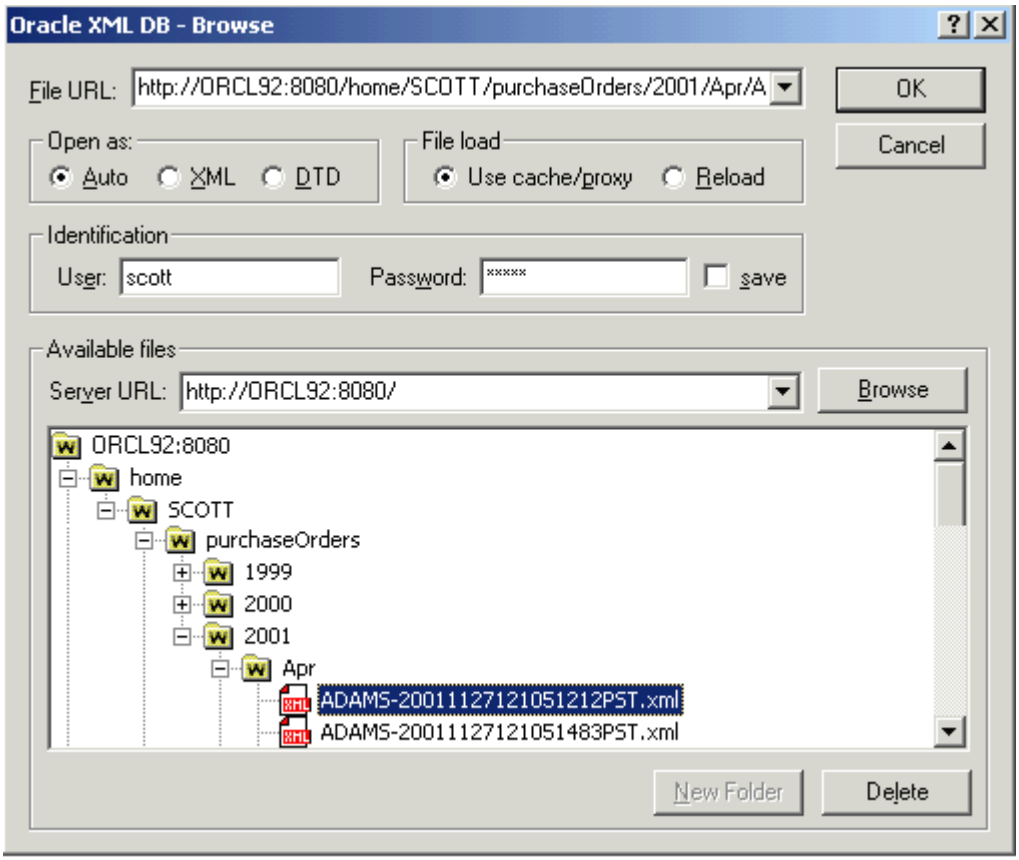
   Hierdurch wird das Dialogfeld Datenverknüpfungseigenschaften geöffnet (oder wählen Sie das Dialogfeld Datenquelle, falls Sie ODBC ausgewählt haben).
4. Wählen Sie **MS OLE DB Provider für Oracle** und klicken Sie auf **Weiter**.
5. Geben Sie die Serverinformationen ein: Server Name, Benutzername und Passwort,

klicken Sie auf das Kontrollkästchen **Speichern des Kennworts zulassen**, und klicken Sie auf OK.
Der ADO Connection String wird nun im Textfeld Connection String angezeigt.

6. Klicken Sie auf **Registrieren**.
Jetzt ist das Schema in der Oracle XML DB registriert.

Oracle XML-Dokumente durchsuchen

Mit diesem Befehl können Sie die XML-Dokumente durchsuchen, die auf Ihrem Server zur Verfügung stehen. Serverinformationen, die von früheren Datenbank-Abfragen zur Verfügung stehen, werden automatisch eingegeben. Sollte dies nicht der Fall sein, geben Sie die Informationen händisch ein.

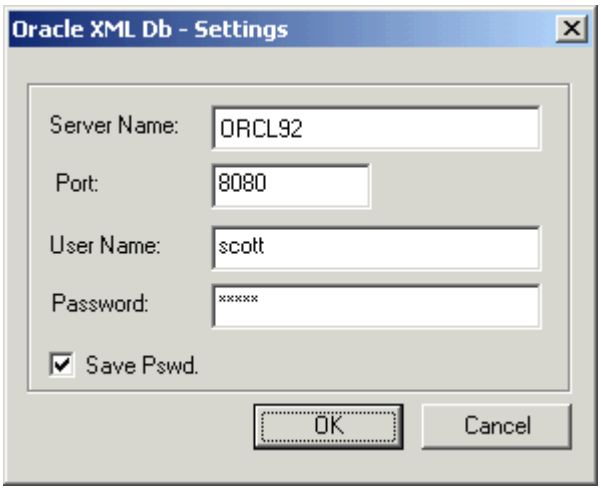


Im Baumdiagramm finden Sie die spezifischen XML Dateien. Doppelklicken Sie auf das Baumdiagramm um es weiter auszuklappen. Sie können auch auf eine Datei doppelklicken und die Schaltfläche OK betätigen. Mit **Neuer Ordner** können Sie neue Ordner einfügen, mit **Löschen** entfernen Sie zuvor ausgewählte XML Dateien.

Eigenschaften

© 2005 Altova GmbH

Mit diesem Befehl zeigen Sie die Eigenschaften der Oracle XML DB für die aktuelle Abfrage, das Schema oder XML-Dokument im Hauptfenster an.



Server-Name
Name des Servers mit dem Sie über http Protokoll verbunden sind.

Port
Serverport, der verwendet wird, um die Serververbindung herzustellen. 8080 ist die localhost-Verbindung.

Benutzername
Ihr Username.

Passwort
Ihr Passwort.

Passw.speichern
Speichert das Passwort für dieses XML-Dokument.

XML Dokumentinstanzen hinzufügen

Hinzufügen einer XML-Dokumentinstanz in die Datenbank:

1. Öffnen Sie ein bereits vorhandenes Schema aus der Oracle XML DB durch Verwendung des Befehls **Schema auflisten**.
2. Wählen Sie die Menüoption **DTD/Schema | XML Beispieldatei erzeugen**.
3. Wählen Sie die spezifischen Optionen für die XML Datei und klicken Sie auf OK. Hierdurch wird die XML Datei erstellt, die mit dem Schema assoziiert ist (bitte lesen Sie die Anmerkung am Ende dieses Kapitels).
4. Wählen Sie die Menüoption **Datei | Speichern unter URL**.
5. Klicken Sie auf die Schaltfläche **Durchsuchen** und navigieren Sie zu dem Ordner, in dem Sie das XML-Instanzdokument speichern möchten. Mit Hilfe der Schaltfläche **Neuer Ordner** können Sie einen neuen Ordner anlegen.
6. Geben Sie den Namen der Datei nach dem Pfad/URL im Feld Datei URL an (bitte stellen Sie sicher, dass die Datei auf dem Oracle Datenbank Server gespeichert wird).
7. Klicken Sie auf die Schaltfläche OK um das XML Instanzdokument an dieser Stelle zu speichern.

Bitte beachten Sie:
Da der Root-Node jedes Dokuments ein **noNamespaceSchemalocation** Attribut

enthält, welches es als Instanz eines registrierten Schemas identifiziert, werden die Dokumente auseinander genommen und als Objekte in der Datenbank gespeichert.

© 2005 Altova GmbH

13.10 Menü "Ansicht"

Im Menü Ansicht kontrollieren Sie die Anzeige für das aktive Hauptfenster. So können Sie sich in XMLSpy 2005 Ihre XML-Dokumente verschiedenartig anzeigen lassen.

In diesem Abschnitt finden Sie eine vollständige Beschreibung aller Befehle im Menü "Ansicht".

13.10.1 Text-Ansicht

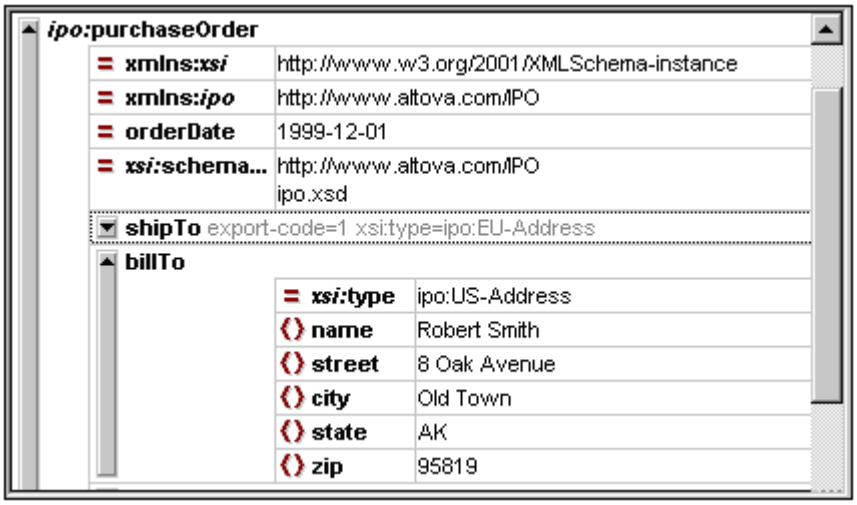
Mit diesem Befehl **wechseln** Sie für das aktuelle Dokument in die Text-Ansicht. In dieser Ansicht können Sie das Dokument in der Textform editieren. Die Text-Ansicht unterstützt eine Reihe von Editierfunktionen, die im Abschnit Text-Ansicht näher beschrieben sind.

Hinweis: Sie können Aspekte der Textansicht mit Hilfe von Optionen konfigurieren, die auf den verschiedenen Registern des Dialogfelds "Optionen" (**Extras | Optionen)** zur Verfügung stehen.

13.10.2 Enhanced Grid Ansicht

Mit diesem Befehl **wechseln** Sie mit dem aktuellen Dokument in die Enhanced Grid Ansicht.

Wenn Sie von der Text-Ansicht in diese Ansicht wechseln, wird Ihr Dokument vorab auf Wohlgeformtheit überprüft.



Eingebettete Datenbank/Tabelle Ansicht
Mit XMLSpy 2005 können Sie sich wiederholende Elemente innerhalb der Enhanced Grid Ansicht in einer Datenbank/Tabelle Ansicht ansehen. Diese Funktion ist immer dann verfügbar, wenn Sie die Enhanced Grid Ansicht aktivieren können, und kann beim Editieren jeder beliebigen XML Datei - XML, XSD, XSL usw. verwendet werden.

Weitere Informationen zu diesem Thema finden Sie auch im Abschnitt Grid-Ansicht.

13.10.3 Schema/ WSDL Design-Ansicht

Mit diesem Befehl wechseln Sie mit dem aktuellen Dokument in die Schema/WSDL Design-Ansicht. Nähere Informationen dazu erhalten Sie im Abschnitt Schema/WSDL Design-Ansicht.

13.10.4 Authentic-Ansicht

Mit diesem Befehl wechseln Sie mit Ihrem Dokument in die Authentic-Ansicht.

Mit der Authentic-Ansicht können Sie XML-Dokumente bearbeiten, die auf in **StyleVision erstellten Vorlagen basieren!** Die Vorlagen werden in StyleVision als Authentic Stylesheet (***.sps** Dateien) gespeichert, und liefern alle für die Authentic-Ansicht benötigten Informationen.

Öffnen Sie die Vorlagen über den Menübefehl **Datei | Neu** und klicken auf die Schaltfläche "StyleVision Stylesheet auswählen...". Weitere Informationen finden Sie in der Dokumentation zur Authentic-Ansicht.

Hinweis: Wenn bei Umschalten in die Authentic-Ansicht eine Meldung angezeigt wird, dass eine temporäre Datei nicht erstellt werden konnte, wenden Sie sich an Ihren Systemadministrator, damit dieser die Standard-Sicherheits-ID für Personen, die nicht als Hauptbenutzer angemeldet sind, ändert, so dass diese Ordner und Dateien erstellen können.

13.10.5 Browser-Ansicht

Mit diesem Befehl wechseln Sie mit dem aktuellen Dokument in die Browser Ansicht.

Diese Ansicht verwendet einen XML-fähigen Browser (am besten Internet Explorer 5) um das XML-Dokument mit den aus CSS oder XSL Style-Sheets zur Verfügung gestellten Informationen darzustellen.

Wenn Sie in die Browser Ansicht wechseln, wird das Dokument zuerst auf Gültigkeit geprüft, wenn Sie die Option "Validieren beim Speichern der Datei" unter **Extras | Optionen** auf dem Register **Datei** einschalten.

Weitere Informationen zu dieser Ansicht finden Sie im Abschnitt Hauptfenster.

13.10.6 Erweitern

Kürzel: **Num +**

Mit diesem Befehl blenden Sie eine weitere Ebene des ausgewählten Elements ein.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden. Tastaturkürzel der Zifferntastatur ist **+**.

In der Enhanced Grid Ansicht, bleiben das Element und seine Child-Elemente nach dem Einblenden markiert. So können Sie große Elemente einblenden indem Sie wiederholt die + Taste betätigen.

© 2005 Altova GmbH

Sie können Elemente auch durch Klicken auf die graue Leiste links eines jeden Elements einblenden oder ausblenden.

13.10.7 Reduzieren

Kürzel: **Num -**

Mit diesem Befehl **blenden** Sie ein Element um eine Ebene **aus**.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden. Tastaturkürzel der Zifferntastatur ist das **-** auf dem **Zahlenblock**.

Sie können Elemente auch durch Klicken auf die graue Leiste links eines jeden Elements einblenden oder ausblenden.

13.10.8 Alles erweitern

Kürzel: **Num ***

Mit diesem Befehl **blenden** Sie alle Child-Objekte des **ausgewählten Elements** ein.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden. Tastaturkürzel der **Zifferntastatur** ist *****.

13.10.9 Unselektierte reduzieren

Kürzel: **Strg Num -**

Mit diesem Befehl können Sie sich auf ein Element und dessen Children konzentrieren und alle anderen Elemente ignorieren.

Der Befehl kann in der Enhanced Grid Ansicht ausgeführt werden. Tastaturkürzel der **Zifferntastatur** ist **Strg und -**.

Wählen Sie einfach das Element aus, mit dem Sie arbeiten möchten und blenden Sie mit diesem Befehl alle anderen nicht selektierten Elemente aus.

13.10.10 Optimale Breite

Mit diesem Befehl stellen Sie die Größe der Spalten so ein, dass der Text des gesamten Dokuments in die zugehörigen Spalten passt.

Wenn Sie mehrere Elemente ein- oder ausblenden, wählen Sie diesen Befehl, da für die Berechnung der optimalen Breite nur sichtbare Objekte in Betracht gezogen werden.

13.10.11 Zeilenumbruch

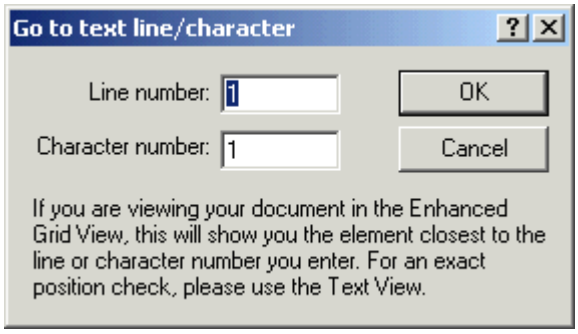
Mit diesem Befehl schalten Sie den Zeilenumbruch in der **Text-Ansicht** ein oder aus.

13.10.12Gehe zu Zeile/Zeichen

Kürzel: **Strg+G**

Mit diesem Befehl gelangen Sie zu einer **spezifischen Zeilennummer** und/oder Zeichenposition in einem XML-Dokument in der Text-Ansicht.

Wenn Sie mit einem externen XSLT Prozessor (weitere Infos: Register XSL unter Extras | Optionen) arbeiten, werden Sie öfter Fehlermeldungen bezüglich Zeilennummer und/oder Zeichenposition erhalten. Mit XMLSpy 2005 können Sie mit diesem Befehl exakt zur gewünschten Position navigieren.



Diese Funktion ist in der Text-Ansicht und Enhanced Grid Ansicht verfügbar, zeigt allerdings in der Grid-Ansicht nur die ungefähre Position an, indem das Element markiert wird, welches sich am nächsten zur gewünschten Position befindet.

13.10.13Gehe zu Datei

Mit diesem Befehl **öffnen** Sie das Dokument, welches auf das von Ihnen momentan bearbeitete Dokument **referenziert**.

Wählen Sie den gewünschten Dateinamen, Pfadnamen, oder die URL und führen Sie diesen Befehl über das Menü Ansicht aus.

Sie können folgendes auswählen:

- Ein vollständiges Element oder Attribut in der Enhanced Grid Ansicht
- Zeichen innerhalb eines beliebigen Elements in der Text oder Enhanced Grid -Ansicht.
- Einen kompletten String. Wenn sich der Cursor zwischen zwei Anführungszeichen befindet, benutzt XMLSpy 2005 automatisch den kompletten String der sich zwischen den Anführungszeichen befindet.

13.10.14Zeilennummerierungsleiste

Mit diesem Befehl wird die Zeilennummerierungsleiste in der Text-Ansicht **ein- oder ausgeblendet**.

© 2005 Altova GmbH

13.10.15Lesezeichenleiste

Mit diesem Befehl wird die Lesezeichenleiste in der Text-Ansicht **ein- oder ausgeblendet**. Wenn in der Lesezeichenleiste ein Lesezeichen vorhanden ist und die Lesenzeichenleiste anschließend ausgeblendet wird, erscheint die mit dem Lesezeichen versehene Zeile markiert. Nach Einblenden der Lesenzeichenleiste wird die Markierung entfernt und Sie sehen darin das Lesezeichen.

13.10.16Klappleiste

Mit diesem Befehl blenden Sie die Klappleiste in der Textansicht ein bzw. aus. Wenn Sie auf der Leiste auf das Plus- bzw. Minus-Symbol klicken wird der entsprechende Code-Abschnitt erweitert (ausgeklappt) bzw. reduziert (eingeklappt).

Bitte beachten Sie:
Sie können Child-Nodes in der Text-Ansicht erweitern oder reduzieren:

Umschalt+Klick auf ein Plus-Symbol **reduziert** alle Child-Nodes dieses Node, während

Strg+Klick auf einen zuvor reduzierten Node alle Child-Nodes **erweitert.**

13.10.17Einrücklinien

Mit diesem Befehl werden die Einrücklinien in der Text-Ansicht ein- oder ausgeblendet. Die Einrücklinien sind vertikale Linien, die entlang des Leerzeichenrandes verlaufen und die hierarchische Ebenenstruktur grafisch verdeutlichen.

13.11 Menü "Browser"

Dieses Menü enthält Befehle, die nur verfügbar sind, wenn die Browser Ansicht aktiviert ist.

13.11.1 Zurück

Taste **Zurück**

Mit diesem Befehl wird eine zuvor angezeigte Seite aufgerufen. Mit der Zurück-Taste auf Ihrer Tastatur erreichen Sie denselben Effekt.
Dieser Befehl ist hilfreich, wenn Sie auf einen Link in Ihrem XML-Dokument klicken und danach zu Ihrem XML-Dokument zurückkehren möchten.

13.11.2 Vorwärts

Dieser Befehl ist nur dann verfügbar, wenn Sie zuvor den Befehl **Zurück** verwendet haben. Mit diesem Befehl gelangen sie vorwärts zur vorher angezeigten Seite.

13.11.3 Abbrechen

Mit diesem Befehl beenden Sie das Laden eines neuen Dokuments.
Dieser Befehl ist nützlich, wenn Sie große externe Dateien oder graphische Darstellungen über eine langsame Internetverbindung laden.

13.11.4 Aktualisieren

F5

Mit diesem Befehl aktualisieren Sie die Browser Ansicht durch erneutes Laden des XML-Dokuments, sowie potentieller externer CSS oder XSL Stylesheets und DTDs.

13.11.5 Schriftgröße

Mit diesem Befehl können Sie die Schriftgröße wählen, die für Text in Ihrem XML-Dokument verwendet werden soll. Die Funktionsweise ist ähnlich wie die, welche in den meisten Browsern zur Anwendung kommt.

13.11.6 Neues Fenster

Mit diesem Befehl öffnen Sie die Browser Ansicht in einem neuen Fenster, sodass Sie Dokumente nebeneinander ansehen können.
Wenn Sie die Browser Ansicht in einem neuen Fenster geöffnet haben, drücken Sie in der Bearbeitungsansicht die Taste **F5**, um die Browser-Ansicht automatisch zu aktualisieren. Um separate Fenster wieder anzudocken, klicken Sie auf die Maximieren-Schaltfläche in der

© 2005 Altova GmbH

rechten oberen Ecke des aktiven Fensters.

13.12 Menü "WSDL"

Dieses Menü enthält Befehle, die nur zur Verfügung stehen, wenn der graphische WSDL Editor aktiviert wurde.

Weitere Informationen finden Sie im WSDL Tutorial und in der Dokumentation zur WSDL Design-Ansicht.

13.12.1 Messages

Neue Message hinzufügen
Eine neue Message wird in der WSDL Datei eingefügt (ersichtlich in der Eingabehilfe-Übersicht). Die Nachricht muss mit einem Input- oder Outputelement assoziiert werden (durch Verwendung von Drag & Drop).

Markierte Message löschen
Löscht die ausgewählte Message aus dem Input- oder Outputelement.

Message Part (Parameter) einfügen
Ein Teil einer Message (Parameter) wird zur ausgewählten Nachricht hinzugefügt.

Message Part (Parameter) löschen
Ein Teil einer Message (Parameter) wird aus der ausgewählten Nachricht gelöscht.

13.12.2 Operations

Operation einfügen
Eine neue Operation wird in das Operationsfenster hinzugefügt.

Operation löschen
Die ausgewählte Operation wird aus dem Operationsfenster gelöscht.

Input-Element einfügen
Ein Input-Element wird in der ausgewählten Operation hinzugefügt.

Output-Element einfügen
Ein Output-Element wird in der ausgewählten Operation hinzugefügt.

Fault-Element einfügen
Ein Fault-Element wird in der ausgewählten Operation hinzugefügt.

Input/Output/Fault-Element löschen
Löscht das ausgewählte Input-, Output- oder Fault-Element.

Neue Message zu diesem Element
Eine neue Message wird zum ausgewählten **Input-** oder **Output-**Element hinzugefügt.

Operation an diesen PortType anhängen
Eine neue Operation wird zum ausgewählten PortType hinzugefügt. Falls das Operations-Fenster leer ist (keine Operationen wurden definiert), kann diese Option nur ausgewählt werden, wenn eines der PortTypes-, Bindings-, oder Servicesfenster aktiv ist.

13.12.3 PortType

PortType einfügen
Ein neuer Porttyp wird im Porttyp-Fenster hinzugefügt.

© 2005 Altova GmbH

PortType löschen
Der ausgewählte Porttyp wird aus dem Porttyp-Fenster gelöscht.

13.12.4 Binding

Neues Binding
Ein neues Binding wird im Bindings-Fenster hinzugefügt.

Binding löschen
Das ausgewählte Binding wird aus dem Bindings-Fenster gelöscht.

Extensibility anhängen
Ein neues Extensibility-Element wird an eine Input- oder Output Message angehängt. Falls ein Menüpunkt an der ausgewählten Stelle nicht angehängt werden darf, ist dieser ausgegraut. Weitere Informationen zu Extensibility-Objekten finden Sie in der W3C WSDL Spezifikation.

Folgende Extensibility-Objekte stehen zur Verfügung:

- soap:body
- soap:fault
- soap:header
- soap:headerfault
- mime:content
- mime:multipartrelated
- mime:part
- mime:mimeXml
- http:urlencoded
- http:urlreplacement
- wsdl:Dokumentation

Extensibility löschen
Löscht das markierte Extensibility-Objekt.

13.12.5 Service

Service einfügen
Ein neuer Service wird im Services-Fenster hinzugefügt.

Service löschen
Der ausgewählte Service wird aus dem Services-Fenster gelöscht.

Port einfügen
Ein neuer Port wird dem ausgewählten Service im Services-Fenster hinzugefügt.

Port löschen
Der ausgewählte Port wird aus dem ausgewählten Service gelöscht.

13.12.6 Types

Neues Schema
Diese Option ist dann verfügbar, wenn die WSDL Datei kein Schema-Element enthält. Bitte beachten Sie, dass bei Aufrufen der Menüoption **Datei | Neu** ein Schemaelement im Skelett der WSDL Datei inkludiert ist. In diesem Fall ist dieser Menüpunkt nicht verfügbar (siehe unten).

```
<types>
        <xs:Schema/>
</types>
```

Schema(s) bearbeiten
Öffnet eine Schemaskelett-Datei, falls die WSDL Datei keine Referenz auf ein spezifisches Schema enthält. Dies ist der Fall, wenn Sie die Menüoption **Datei | Neu** verwendet haben.

Wenn eine Referenz zu einem spezifischen Schema existiert, wird das Schema in der eingebetteten Schema-Ansicht des graphischen WSDL Editors angezeigt.

13.12.7 Schema-Dokumentation generieren

Erzeugt die Dokumentation für die aktuelle WSDL Datei. Die Ausgabe der Dokumentation kann als MS Word oder HTML-Datei erfolgen. Sie können angeben, welche Komponenten inkludiert werden sollen. Verwandte Elemente sind miteinander über Links verknüpft, sodass eine einfache Navigation gewährleistet ist.

13.12.8 Einstellungen

Mehrfarbige Verbindungslinien
Mit dieser Funktion können Sie die Farbe der Verbindungslinien zwischen farbig und schwarz/weiß wählen.

© 2005 Altova GmbH

13.13 Menü "Soap"

XMLSpy 2005 unterstützt SOAP Version 1.1 und WSDL Version 1.1.

Bitte beachten Sie, dass bei der Erzeugung von SOAP-Requests das sichere http-Protokoll **HTTPS** nicht unterstützt wird.

Im Abschnitt **Soap Debugger**, der auf die Menübeschreibungen folgt, wird anhand des **nanonull.com timeservice Server**, der Ihnen von Altova zur Verfügung gestellt wird, die Funktionsweise des SOAP Debuggers beschrieben. Bitte verwenden Sie diesen Service, um den Debugger zu testen. Der AirportWeather Webservice, der auf den folgenden Seiten beschrieben wird, steht möglicherweise nicht immer zur Verfügung.

Verwenden Sie die SOAP Funktionalitäten:

- zum Testen Ihrer Webservices, ohne dass Client-Applikationen implementiert werden müssen
- zum schnellen Testen von Webservices von anderen Anbietern

Weitere Informationen zu den Spezifikationen erhalten Sie unter:
SOAP http://www.w3.org/TR/SOAP/
WSDL http://www.w3.org/TR/wsdl

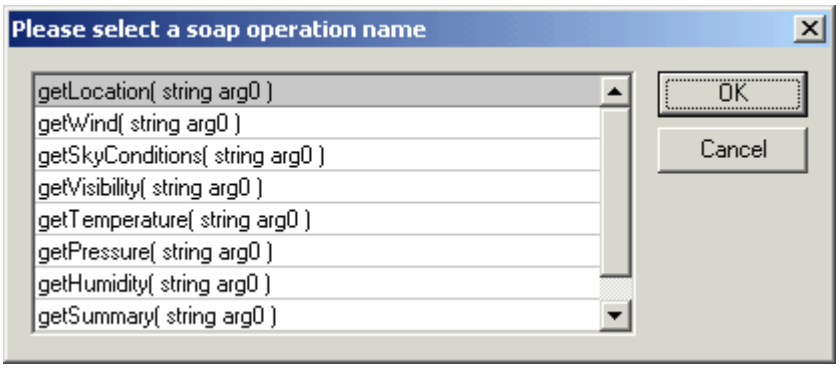
13.13.1 SOAP Request erzeugen

Mit diesem Befehl erzeugen Sie ein neues SOAP Request-Dokument.

Erzeugen eines neuen SOAP Requests:

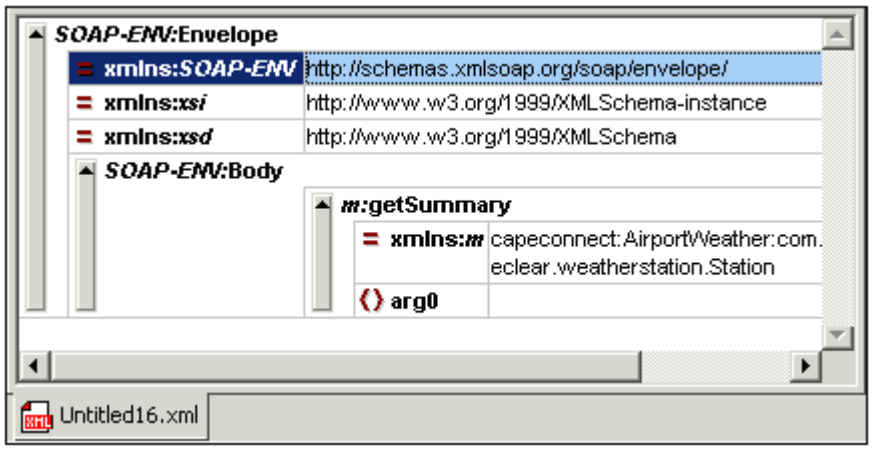
1. Wählen Sie die Menüoption **Soap | SOAP Request erzeugen**.
   Im Dialogfeld geben Sie den lokalen Pfad oder die URL einer WSDL Datei ein, die den Webservice beschreibt. In diesem Beispiel verwenden Sie bitte capescience.com AirportWeather Webservice. Die URL für diesen Webservice lautet:
   http://www.capescience.com/webservices/airportweather/AirportWeather.wsdl

2. Klicken Sie auf OK um zu bestätigen und zum Herstellen der Verbindung zum capescience.com Server.
   Ein Dialogfeld wird angezeigt, in dem Sie eine bestimmte SOAP Operation auswählen können.

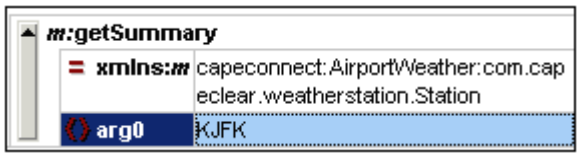


Bitte beachten Sie:
Sollte die Verbindung nicht zustande kommen, oder die URL nicht korrekt sein, wird an dieser Stelle eine Fehlermeldung angezeigt.

3. Klicken Sie auf den Eintrag **getSummary (string arg0)** und bestätigen Sie mit OK. Ein Untitled.XML-Dokument wird in XMLSpy 2005 geöffnet, welches die SOAP-Request Daten enthält.



4. Klicken Sie auf eines der Felder zum Abwählen der Markierung der Tabelle. Bevor der Request abgesendet werden kann, müssen Sie angeben, für welchen Flugplatz Sie eine Wettervorhersage benötigen.
5. Klicken Sie auf das Feld **arg0** und geben Sie **KJFK** für den Flughafen New York JFK Airport ein.

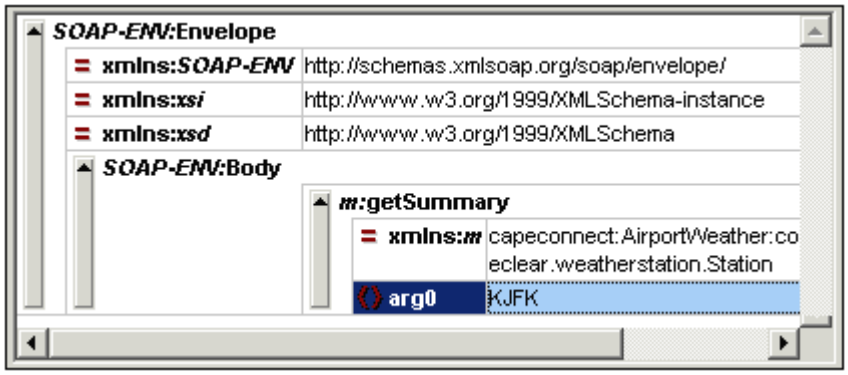


Nun ist die **Definition** des SOAP Requests vollständig und wird versendet. Jetzt müssen wir nur noch warten, ob wir eine Antwort vom Server erhalten. Siehe auch "Request an Server senden".

13.13.2 SOAP Request an Server senden

Mit diesem Befehl wird der SOAP Request über XMLSpy 2005 an den Webservice gesendet. Wenn Sie ein SOAP Request Dokument erzeugt haben, (siehe hierzu Neuen SOAP Request erzeugen), wird der Request über diesen Befehl versendet.

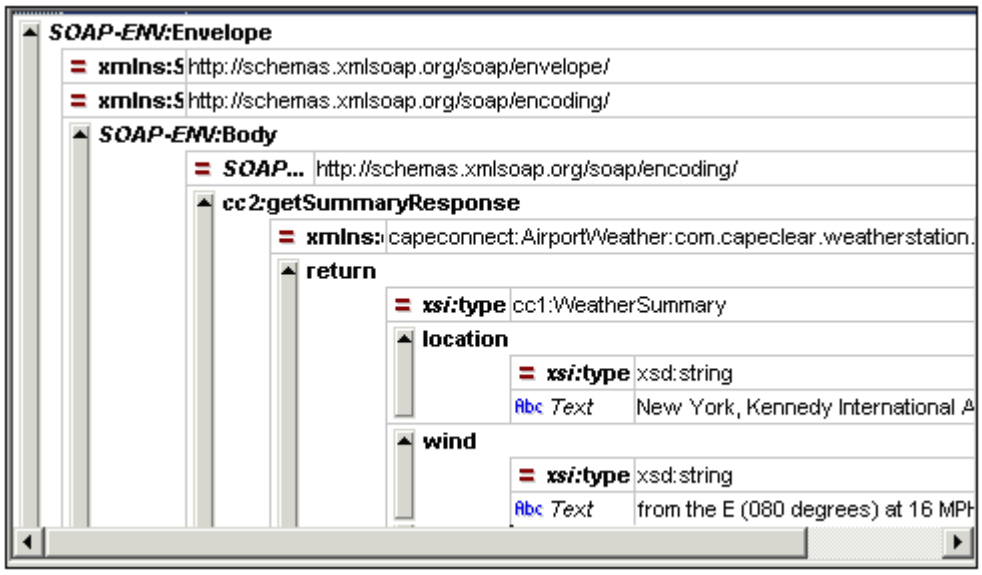
Zuvor definiertes SOAP Request Dokument:



So versenden Sie einen SOAP Request:

1. Stellen Sie sicher, dass das SOAP Request Dokument aktiv ist, und wählen Sie die Menüoption **Soap | SOAP Request an Server senden**.

   Nach einigen Sekunden wird automatisch ein neues XML-Dokument erstellt, welches den SOAP Response vom zuständigen Server enthält. In diesem Beispiel werden im neuen Dokument Standort und Wetterinformationen für den Flughafen Neu York JFK dargestellt.



Um die SOAP Request Parameter einzustellen, die nicht im Request Dokument angezeigt werden (da diese im HTTP Header transferiert werden), verweisen wir auf "SOAP Request Parameter ändern".

Speichern und erneute Verwendung eines SOAP Request:

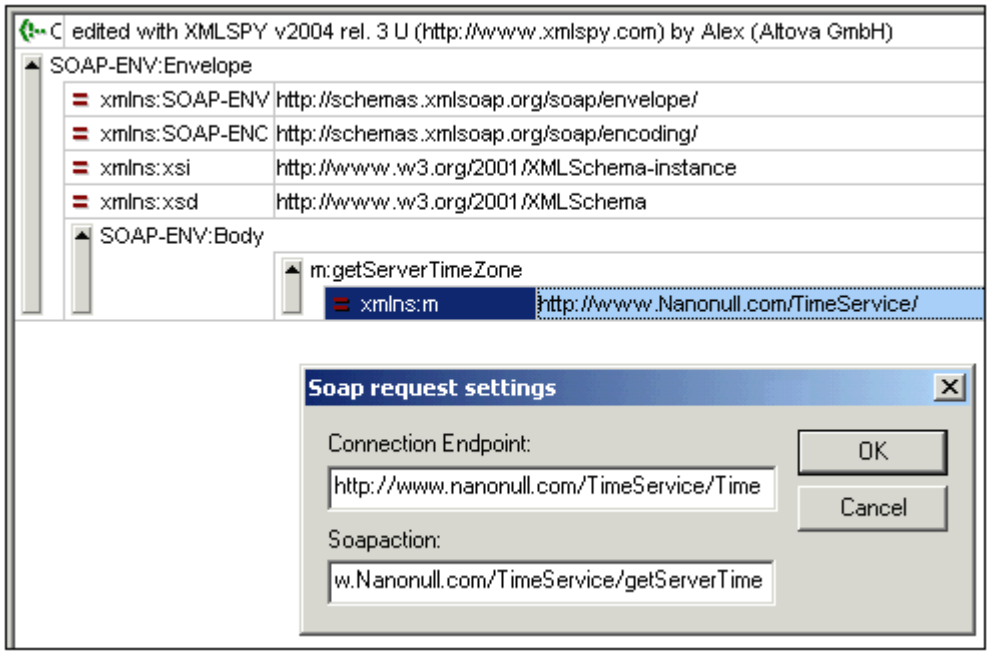
In XMLSpy 2005 können Sie einen SOAP Request speichern und zu einem späteren Zeitpunkt erneut senden:

Nach Erzeugen eines SOAP Request mittels der Option **SOAP Request erzeugen**

1. Wählen Sie die Menüoption **Datei | Speichern unter...** und benennen Sie das SOAP Request XML-Dokument.
   Schließen Sie die SOAP Request-Datei.
2. Öffnen Sie das SOAP Request XML-Dokument erneut und wählen Sie die Menüoption **SOAP | SOAP Request-Parameter ändern.**
3. Daraufhin werden Sie gefragt, ob Sie diese Datei als SOAP Request definieren wollen und informiert, dass Sie die SOAP Request Parameter manuell eingeben müssen. (Als SOAP Request-Dokument kann jede XML-Datei verwendet werden).

4. Klicken Sie auf **Ja** und geben Sie den **Connection Endpoint** und die **SOAP Operation** in die entsprechenden Felder ein.
5. Wählen Sie die Menüoption **SOAP | SOAP Request an Server senden,** um den Request abzusenden.

Bitte beachten Sie:
Diese beiden Parameter **werden nicht gespeichert**, wenn Sie das SOAP Request-Dokument speichern. Sie müssen sie manuell eingeben, wenn Sie das SOAP Request-Dokument erneut öffnen.

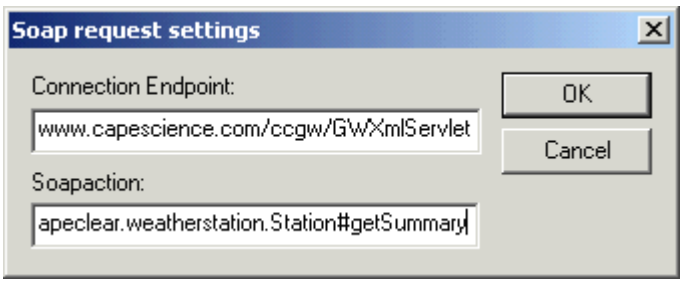


13.13.3 SOAP Request-Parameter ändern

Mit diesem Befehl können Sie die aktuell gültigen SOAP Parameter des SOAP Request Dokuments ändern. Das vorliegende Beispiel verwendet das zuvor definierte SOAP Request Dokument.

So ändern Sie die Parameter für den SOAP Request:

1. Stellen Sie sicher, dass das SOAP Request Dokument aktiv ist und wählen Sie die Menüoption **Soap | SOAP Request-Parameter ändern**.
   Das Dialogfeld "SOAP Request-Einstellungen" wird geöffnet. Sie können nun den Connection Endpoint sowie die SOAP Operation ändern. Bestätigen Sie Ihre Änderungen mit OK.



2. Sie können nun den SOAP Request erneut senden (Menüoption **Soap | SOAP**

 © 2005 Altova GmbH

Request an Server senden).
Ein SOAP Response Dokument wird erzeugt und nach einigen Sekunden im Hauptfenster dargestellt.

Bitte beachten Sie:
Ändern Sie die Soap Operation-Einstellungen nur, wenn Ihnen mehrere Aktionen in einer Liste zur Verfügung stehen.

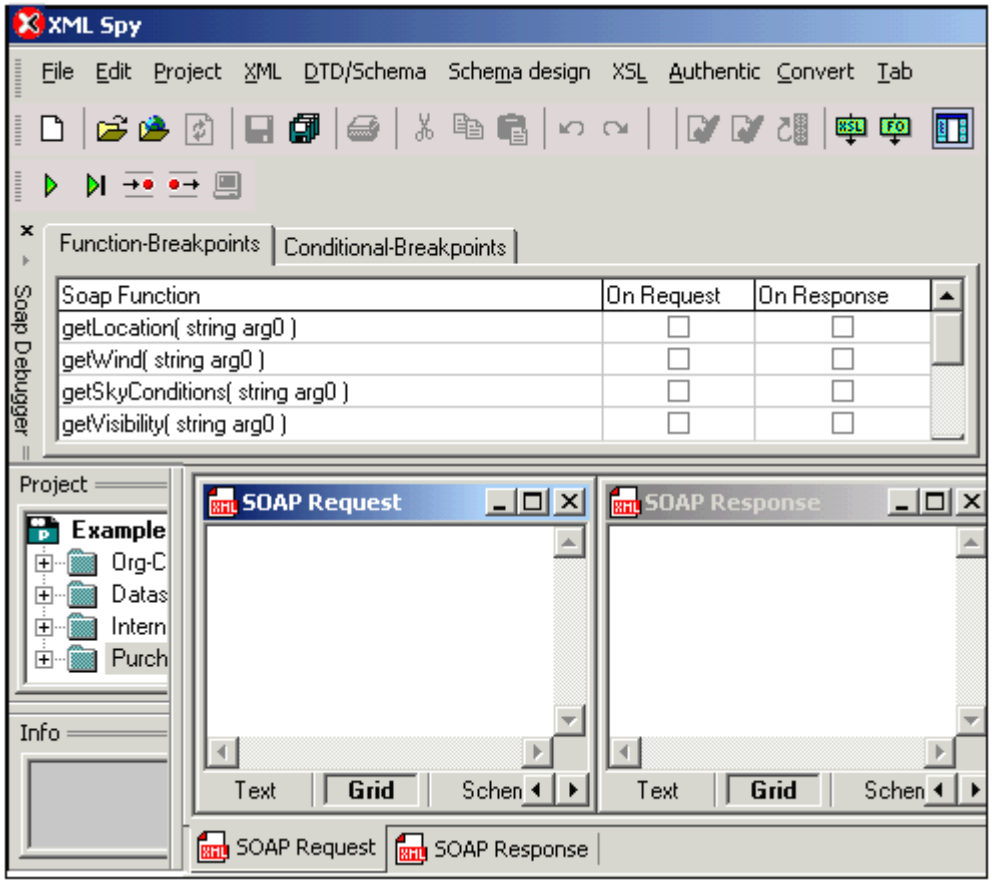
13.13.4 Soap Debugger-Sitzung

Mit diesem Befehl starten Sie eine SOAP Debugger-Sitzung.

- Wenn Sie diesen Befehl ausführen, wird ein Dialogfeld angezeigt. Wählen Sie den Standort der WSDL-Datei (normalerweise eine URL).
- Definieren Sie im nächsten Dialogfeld die Quell- und Ziel-Ports, die für den Debugger Proxyserver und das Webservice benötigt werden.

  Der SOAP Debugger Proxyserver wird geöffnet, ist aber **noch nicht aktiv**. Klicken Sie auf eines der Symbole in der Symbolleiste. Der SOAP Debugger wird gestartet und wartet nun auf Ihre Befehle.

  Weitere Informationen erhalten Sie unter SOAP Debugger .



13.13.5 Starten

Mit diesem Befehl wird der SOAP Proxy Server gestartet. Die Prozedur wird solange in der WSDL-Datei fortgesetzt, bis ein Breakpoint erreicht wird. Das zugehörige SOAP Dokument wird dann in einem der SOAP Dokument-Fenster angezeigt.

13.13.6 Einzelschritt

Mit diesem Befehl können Sie sich schrittweise durch ein- und ausgehende SOAP Requests und Responses bewegen. Bei jedem Request und Response hält der Debugger automatisch an. Der Proxyserver wird auch gestartet, falls dieser inaktiv war.

13.13.7 Beim nächsten Request anhalten

Mit diesem Befehl hält der Debugger beim nächsten SOAP Request an und die Daten werden im SOAP Request Dokument-Fenster dargestellt. Sie können die Daten im Fenster bearbeiten bevor Sie diese an den Webservice senden.

13.13.8 Bei nächstem Response anhalten

Mit diesem Befehl hält der Debugger beim nächsten SOAP Response an und die Daten werden im SOAP Response Dokument Fenster dargestellt. Sie können die Daten im Fenster bearbeiten bevor Sie diese an den Client senden.

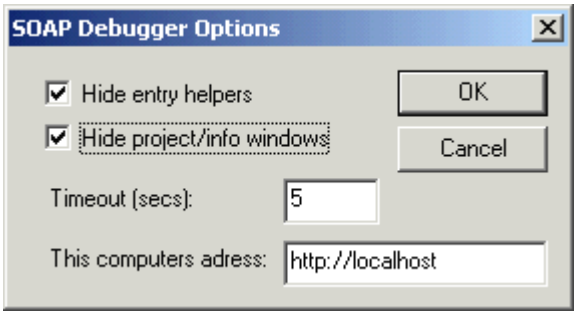
13.13.9 Proxyserver anhalten

Mit diesem Befehl halten Sie den Debugger Proxyserver an.

13.13.10 Soap Debugger Einstellungen

Mit diesem Befehl definieren Sie die Optionen für den SOAP Debugger. Bitte definieren Sie die Optionen **bevor** Sie den Debugger starten.

Ausblenden:
Sie können sowohl die Eingabehilfen als auch das Projektfenster ausblenden, um Ihr SOAP Dokument besser ansehen zu können.



 © 2005 Altova GmbH

Timeout (sek.)
Die meisten Clients haben eine Timeout-Einstellung für das Einlangen von Daten. Die Standardeinstellung der Zeit, die der Debugger bei dem jeweiligen Breakpoint bleibt, ist bei 5 Sekunden. Eine Datenpaketnachricht (http1.1 100 Continue) wird nach jeder von Ihnen definierten Timeoutperiode an den Client gesendet.

Wenn Sie die Timeoutperiode auf Null setzen, bleiben Sie bei dem Breakpoint, solange Sie wollen.

Adresse des Rechners:
Adresse des Computers, auf dem der SOAP Debugger läuft. Dieser Computer muss für den Client-Computer zugänglich sein.

Geben Sie die Adresse in folgender Form ein: **http://computer Name**, wobei "Computer Name" für die Webadresse oder IP-Adresse des Computers steht, auf dem der SOAP Debugger läuft. Geben Sie in der Adresse keine Port-Nummer an. Der Eintrag "localhost" verweist immer auf Ihren lokalen Rechner.

13.14 Menü "Extras"

Im Menü Extras können Sie:

- Die Rechtschreibung Ihrer XML-Dokumente überprüfen
- In die Skript-Umgebung von XMLSpy 2005 wechseln. Sie können Ihre eigenen Formulare, Makros und Event Handler erstellen, bearbeiten und speichern.
- Derzeit zugewiesene Makros ansehen
- Skripts einem Projekt zuweisen oder zugewiesene Skripts löschen
- zwei beliebige Dateien vergleichen und auf Unterschiede überprüfen
- zwei beliebige Ordner vergleichen und auf Unterschiede überprüfen
- Ihre Version anpassen: eigene Menüleisten, Tastaturkürzel, Menüs und Makros erstellen
- Globale Programm-Einstellungen definieren

13.14.1 Rechtschreibung...

XMLSpy 2005 enthält jetzt auch eine integrierte Rechtschreibprüfung.

Die Rechtschreibprüfung kann in folgenden Ansichten verwendet werden: Enhanced Grid, Text und Authentic, sowie in StyleVision. Bitte beachten Sie, dass jede Art von XML Datei überprüft werden kann. Die Ansicht, in der Sie die Rechtschreibung überprüfen, beeinflusst die Optionen, die für die Rechtschreibprüfung zur Verfügung stehen. Eine Schemadatei (*.xsd) kann in der Text und Enhanced Grid Ansicht, aber nicht in der Schema/WSDL Design-Ansicht überprüft werden.

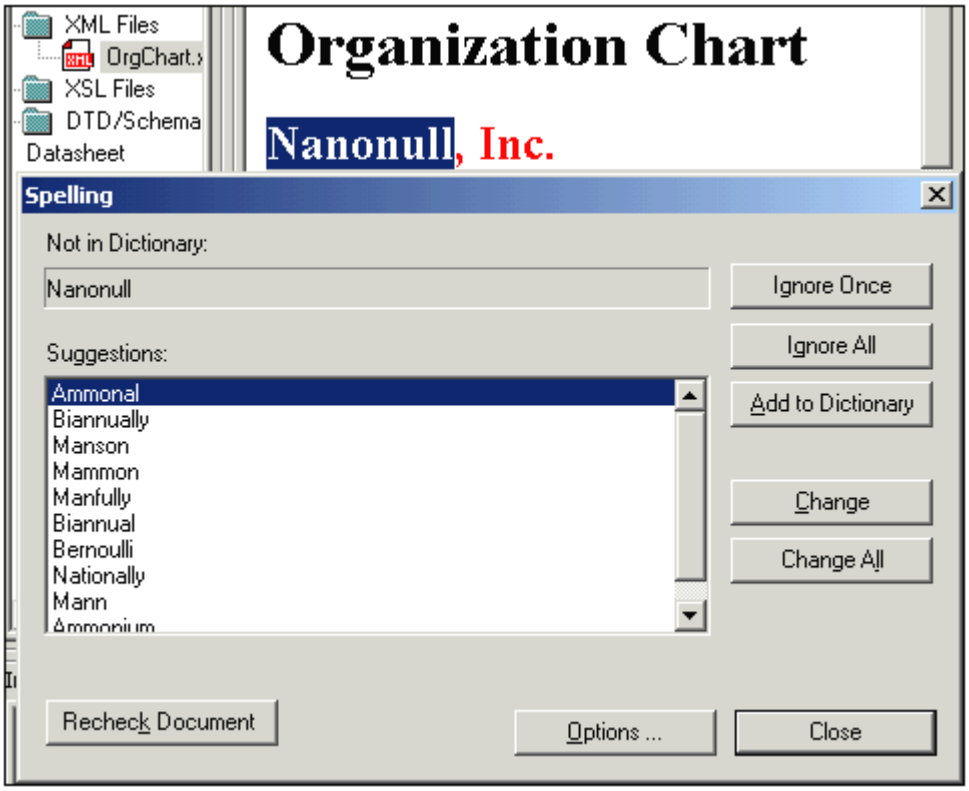
Die Rechtschreibprüfung in der **Text-** und **Enhanced Grid-Ansicht** bietet Ihnen die meisten Optionen für die Überprüfung. Sie können zusätzlich die Rechtschreibung von folgenden Objekten überprüfen:

- Inhalt von Elementen
- Inhalt von Attributen
- Inhalt von CDaten
- Kommentartexten

  Weiters können Sie definieren, welche Inhalte von Elementen oder Attributen überprüft oder übersprungen werden sollen. Die Einstellungen werden im Dialogfeld "Rechtschreibkontext" vorgenommen.

Mit dem Befehl **Extras | Rechtschreibung...** (Umschalt+F7) öffnen Sie das Dialogfeld für die Rechtschreibprüfung. Das aktive XML-Dokument wird automatisch auf Rechtschreibfehler geprüft (in diesem Fall, das OrgChart.XML-Dokument in der Authentic-Ansicht).

© 2005 Altova GmbH



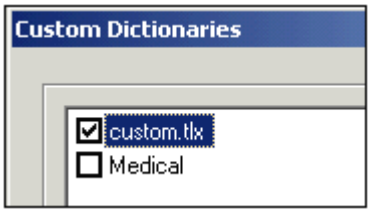
Nicht im Wörterbuch
Dieses Textfeld enthält ein Wort aus dem Text, welches in keinem der existierenden Wörterbücher (Standardwörterbuch und benutzerdefinierte Wörterbücher) vorkommt. In diesem Textfeld können Sie auch direkt Text bearbeiten. Das Wort wird im XML-Dokument zur besseren Übersichtlichkeit markiert anzeigt. Die Schaltflächen werden nun aktiviert, und Sie können entscheiden, wie Sie fortfahren möchten.

Vorschläge
Dieses Listenfeld zeigt eine Reihe von Wörtern an, die dem unbekannten Wort am nähesten kommen (aus allen verfügbaren Wörterbüchern). Durch Doppelklicken auf einen Eintrag wird dieser anstelle des unbekannten Wortes eingefügt und die Rechtschreibprüfung wird fortgesetzt.

Einmal ignorieren
Mit diesem Befehl können Sie das unbekannte Wort bei seinem ersten Auftreten ignorieren und die Rechtschreibprüfung fortsetzen. Sollte das Wort ein weiteres Mal im Dokument vorkommen, wird die Rechtschreibprüfung es dann wieder als unbekannt markieren.

Immer ignorieren
Mit diesem Befehl wird das Wort durch das ganze Dokument ignoriert.

Ins Wörterbuch
Mit diesem Befehl können Sie unbekannte Wörter in das aktive benutzerdefinierte Wörterbuch einfügen. Das aktive benutzerdefinierte Wörterbuch (custom.tlx) ist im Dialogfeld "Benutzerdefinierte Wörterbücher" mit einem Häkchen versehen.



Ändern
Mit diesem Befehl ersetzen Sie das im XML-Dokument markierte Wort durch den markierten Eintrag aus der Auswahlliste der Vorschläge.

Alle ändern
Mit diesem Befehl ersetzen Sie das Wort an allen Stellen im XML-Dokument, an denen es vorkommt.

Dokument erneut kontrollieren
Das Dokument wird vom Anfang an neu überprüft.

Schließen
Mit diesem Befehl beenden Sie die Rechtschreibprüfung.

Einstellungen...
Mit diesem Befehl öffnen Sie ein Dialogfeld in der aktuellen Ansicht:

- In der Text oder Enhanced Grid-Ansicht und in der Authentic-Ansicht wird das Dialogfeld Rechtschreibkontext geöffnet.

13.14.2 Rechtschreibung - Optionen...

Mit diesem Befehl öffnen Sie das Dialogfeld **Optionen** oder **Rechtschreibkontext**, je nachdem, in welcher Ansicht Sie Ihr XML-Dokument derzeit bearbeiten.

Optionen - **Authentic-Ansicht** und **StyleVision**
Bei der Ansicht eines XML-Dokuments in der Authentic-Ansicht wird das Dialogfeld "Optionen" geöffnet. Verwenden Sie dieses Dialogfeld um Optionen für die Rechtschreibprüfung zu definieren.

Korrektur immer vorschlagen:
Wenn Sie dieses Kontrollkästchen aktivieren, werden alle Vorschläge (aus allen Wörterbüchern) in der Vorschlagsliste angezeigt. Wenn Sie diese Option deaktivieren, werden keinen Vorschläge angezeigt.

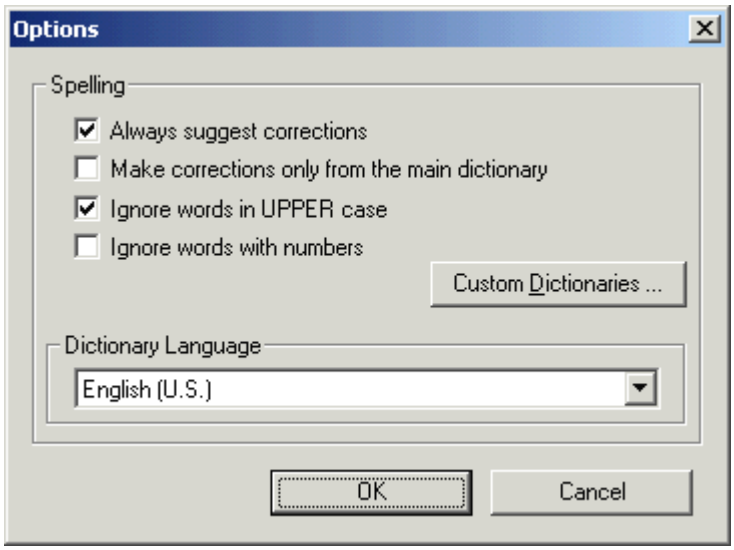
Korrekturen nur aus dem Hauptwörterbuch übernehmen:
Bei Aktivierung dieser Option werden nur Vorschläge angezeigt, die im Hauptwörterbuch zur Verfügung stehen. Weiters wird die Schaltfläche "Benutzerdefinierte Wörterbücher" deaktiviert.

Großschreibung ignorieren:
Großgeschriebene Wörter werden in der Rechtschreibprüfung ignoriert.

Wörter mit Ziffern ignorieren:
Wörter, die Ziffern enthalten, werden in der Rechtschreibprüfung ignoriert.

Sprache Wörterbuch
Wählen Sie eine Sprache im Dropdown-Listenfeld, in der die Rechtschreibprüfung erfolgen soll. Die standardmäßig ausgewählte Sprache ist derzeit Englisch. Weitere Sprachen werden demnächst zum Download von Altovas Webseite zur Verfügung stehen.

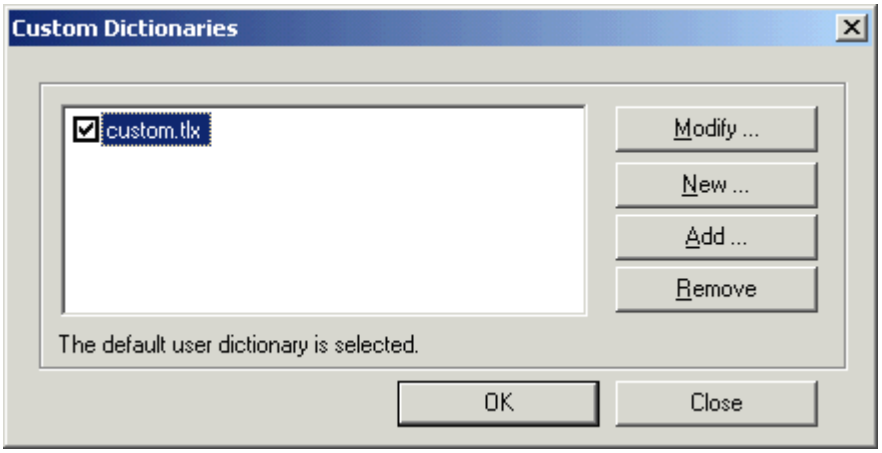
© 2005 Altova GmbH



Benutzerdefinierte Wörterbücher
Mit dieser Schaltfläche können Sie:

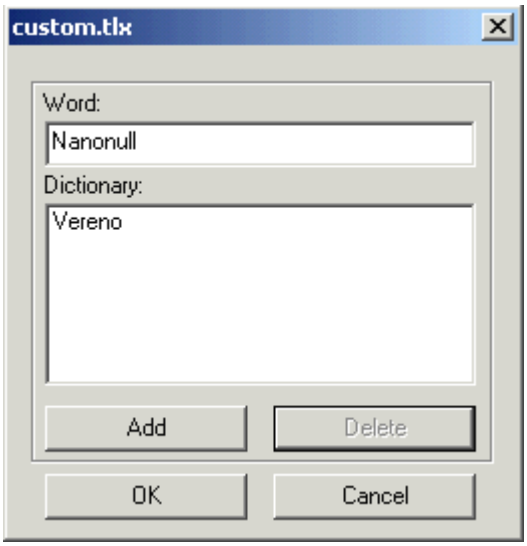
- existierende Wörterbücher bearbeiten (Einträge hinzufügen oder löschen)
- neue Wörterbücher erstellen
- ein existierendes Wörterbuch hinzufügen
- ein existierendes Wörterbuch löschen

Wenn Sie die Rechtschreibprüfung starten, werden alle im Listenfeld angezeigten benutzerdefinierten Wörterbücher durchsucht. Mit dem Befehl Entfernen können Sie bestimmte Wörterbücher deaktivieren. Diese werden während der Rechtschreibprüfung nicht verwendet.



So ändern Sie einen Wörterbuch-Eintrag:

1. Klicken Sie auf den Namen des benutzerdefinierten Wörterbuches, dessen Einträge Sie ändern möchten, und klicken Sie auf Ändern...
   Das ausgewählte Wörterbuch wird geöffnet (in unserem Beispiel, custom.tlx). Falls Sie kein Wörterbuch ausgewählt haben, werden Sie dazu aufgefordert.



2. Klicken Sie auf das Feld "Wort:" und geben Sie den neuen Wörterbucheintrag ein (die Schaltfläche "Hinzufügen" wird hierdurch automatisch aktiviert).
3. Klicken Sie auf "Hinzufügen", um den Eintrag in das Wörterbuch einzufügen.

So löschen Sie einen Eintrag aus dem Wörterbuch:

1. Klicken Sie auf das zu löschende Wort im Listenfeld, und klicken Sie auf Löschen.
2. Klicken Sie auf OK um Ihre Änderungen zu bestätigen.

So fügen Sie ein neues Wörterbuch ein:

1. Klicken Sie auf "Neu", und geben Sie den Namen des neuen benutzerdefinierten Wörterbuches im Feld "Dateiname..." ein.
2. Klicken Sie auf Speichern um das Wörterbuch zu speichern.
3. Über die Schaltfläche "Hinzufügen" oder "Ins Wörterbuch" können Sie nun während der Rechtschreibprüfung neue Einträge in das neue Wörterbuch einfügen.

So fügen Sie ein existierendes Wörterbuch ein:

Verwenden Sie diese Option um zuvor deaktivierte Wörterbücher (oder Wörterbücher von anderen Anbietern) wieder einzufügen.

1. Klicken Sie auf "Hinzufügen" und wählen Sie das Wörterbuch aus der Liste aus. Wörterbücher erkennen Sie an der Erweiterung *.**tlx**.
2. Klicken Sie auf "Öffnen", um das Wörterbuch zu aktivieren.

Bitte beachten Sie:
Wörterbücher müssen nicht unbedingt eine *.tlx Erweiterung haben. Das Wörterbuch kann eine beliebige oder auch keine Erweiterung haben und trotzdem in die Liste der Wörterbücher aufgenommen werden.

So entfernen Sie ein Wörterbuch:

- Klicken Sie auf den Namen des Wörterbuchs und dann die Schaltfläche "Entfernen". Das Wörterbuch wird aus der Liste, aber nicht von der Festplatte gelöscht.

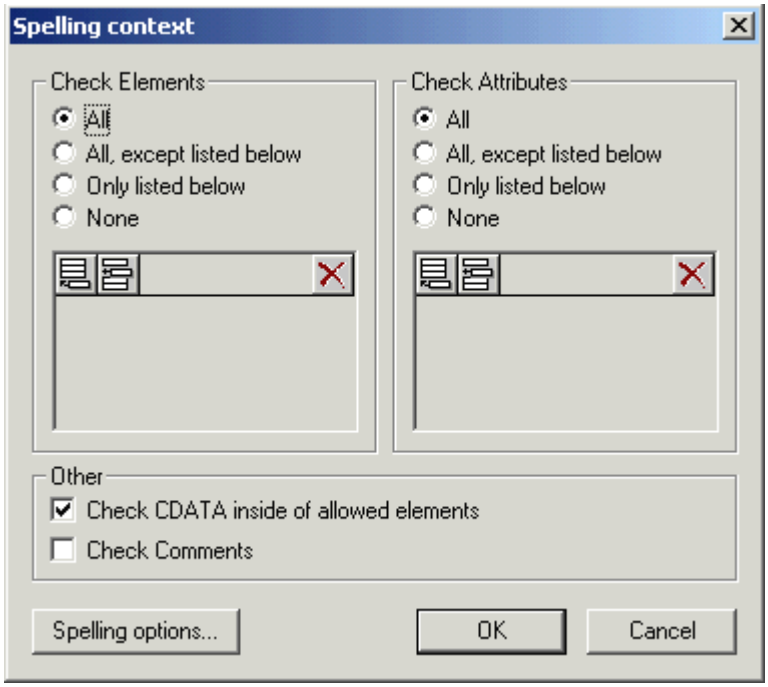
Rechtschreibkontext - **Text-**und **Enhanced Grid-Ansicht**

Wenn Sie Ihr XML-Dokument in der Text- oder Enhanced Grid-Ansicht bearbeiten, können Sie mit diesem Befehl das Dialogfeld "Rechtschreibkontext" öffnen. Wenn Sie die Rechtschreibung in der **Text-** und **Enhanced Grid-Ansicht** starten, haben Sie die meisten Optionen zur Verfügung. So können Sie zum Beispiel folgende Inhalte auf Rechtschreibung überprüfen:

- Inhalte von Elementen

© 2005 Altova GmbH

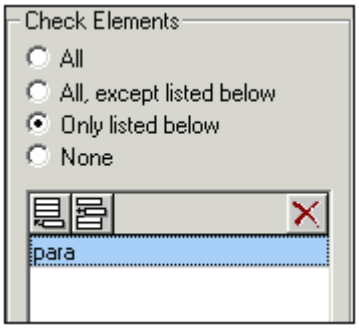
- Inhalte von Attributen
- Inhalte von CDaten
- Kommentartexte



Die Schaltfläche Rechtschreiboptionen... öffnet das Dialogfeld Einstellungen, wo Sie Einstellungen für die Rechschreibprüfung definieren können.

So definieren Sie, welcher Inhalt von Attributen oder Elementen überprüft werden soll:

1. Klicken Sie auf das Optionsfeld "Alle außer unten aufgelistete" oder "Nur unten aufgelistete".
2. Klicken Sie auf das Symbol "Element/Attribut anhängen" , und geben sie den Tag-Namen ein, dessen Inhalt überprüft werden soll (z.B. para). Bestätigen Sie mit der Eingabetaste.



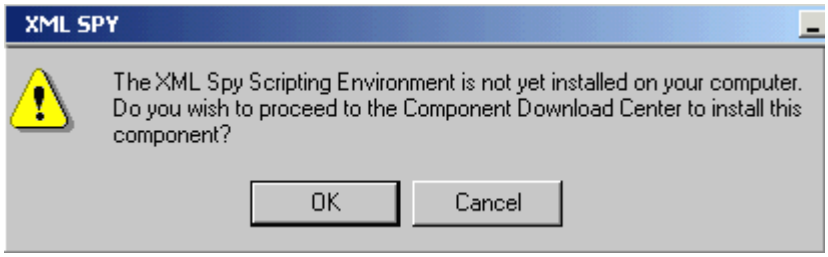
3. Klicken Sie auf die OK Schaltfläche um Ihre Einstellungen zu bestätigen, und bestätigen Sie diese nochmals im Dialogfeld "Rechtschreibung" um die Rechtschreibprüfung mit den neuen Einstellungen zu starten.
   In diesem Fall wird **nur der Text innerhalb der Para-Tags**, sowie dessen Attribute auf Rechtschreibfehler überprüft.

So löschen Sie einen Tag-Namen:

- Wählen Sie den Tag-Namen, den Sie löschen möchten, und drücken Sie die Entfernen- oder die Rücktaste Ihrer Tastatur.

13.14.3 Skripting-Umgebung

Die Skripting-Umgebung ist derzeit eine downloadbare Komponente. Wenn Sie die Komponente noch nicht heruntergeladen und installiert haben, werden Sie dazu aufgefordert, sobald Sie diesen Menübefehl wählen.



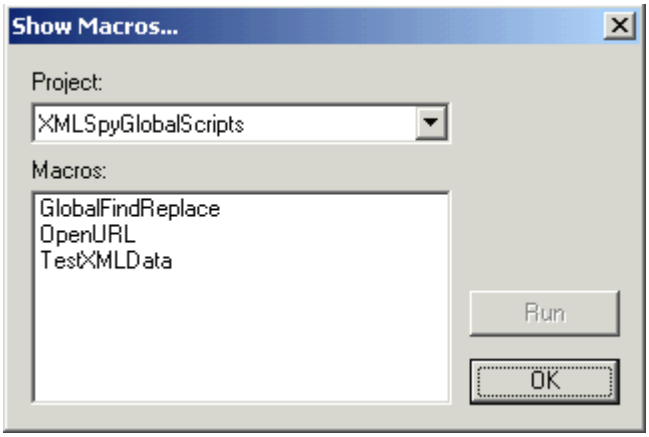
Klicken Sie auf OK um mit dem Download-Center auf Altovas Webseite zu verbunden zu werden. Hier wählen Sie dann die Komponente Skript-Umgebung.

Nachdem Sie die Komponente heruntergeladen haben, doppelklicken Sie auf **Spyscripting.exe** um diese zu installieren.

Die Scripting-Umgebung, ist ein vordefiniertes globales Skript-Projekt, und wird beim ersten Starten des Formular-Editors aktiviert. Weitere Informationen zu diesem Thema finden Sie im Skript-Abschnitt der vorliegenden Dokumentation.

13.14.4 Makros...

Mit diesem Befehl wird das Dialogfeld mit der Liste aller Makros angezeigt, die mit dem globalen Skriptprojekt und mit dem aktuellen Projekt in XMLSpy 2005 assoziiert sind.



Die Schaltfläche "Starten" im Dialogfeld ruft das markierte Makro auf.

13.14.5 Projekt

Mit diesem Befehl können Sie Skripts einem bestimmten Projekt zuweisen (oder löschen).

Bitte beachten Sie:
Diese Menüoptionen sind nur aufrufbar, wenn Sie ein Projekt in XMLSpy 2005 geöffnet

 © 2005 Altova GmbH

haben.

Skripts zuweisen

Mit diesem Befehl können Sie einem Skiptprojekt ein Skript zuweisen.

Projekt-Skripts entkoppeln

Mit diesem Befehl können Sie in einem Skript-Projekt eine Skirpt-Zuweisung löschen.

Projekt-Skripts aktivieren

Mit diesem Befehl können Sie zeitweilig ein Projekt-Skript aktivieren oder deaktivieren.

13.14.6 Vergleiche

XMLSpy 2005 bietet eine Vergleichs- (oder Differencing)-Funktionalität, mit der XML- und Textdateien sowie Ordner auf Unterschiede überprüft werden können.

Das Menü Extras enthält drei Menüpunkte, mit Hilfe derer Sie Vergleiche an Dateien und Ordnern durch führen können:

- Geöffnete Datei vergleichen mit
- Verzeichnisse vergleichen
- Vergleichseinstellungen

Diese Befehle werden in den folgenden Unterabschnitten näher beschrieben.

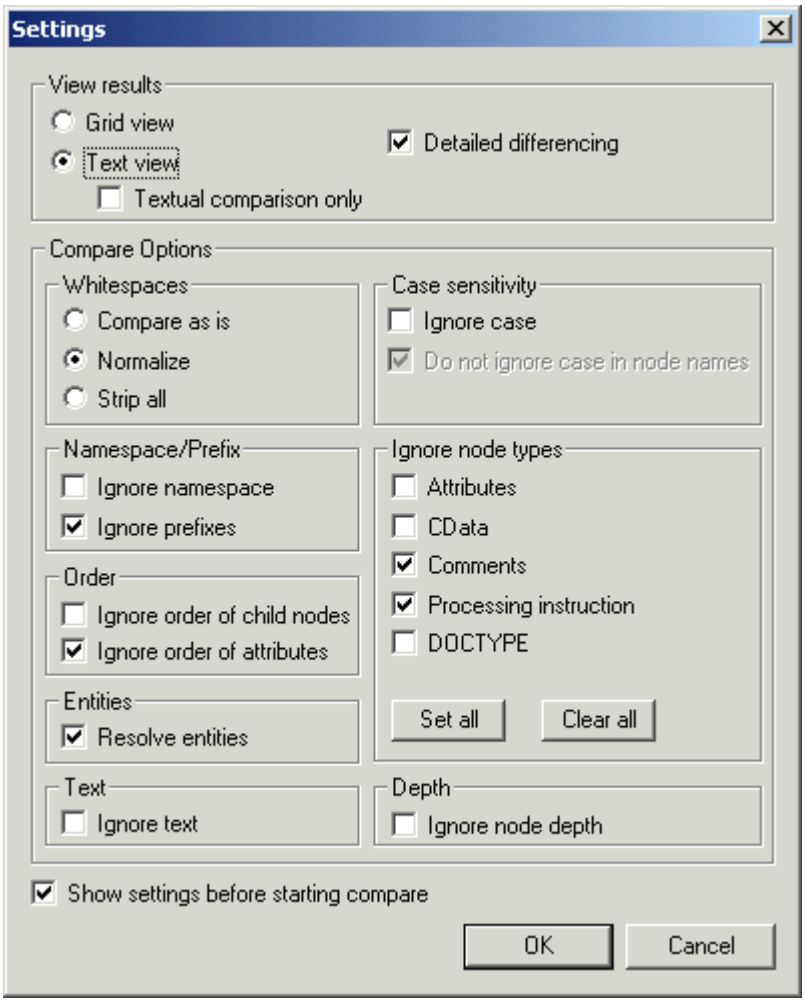
Geöffnete Datei vergleichen mit...

Mit diesem Befehl können Sie eine geöffnete Datei mit einer anderen Datei vergleichen. Beim Vergleich werden beide Dateien im Hauptfenster senkrecht nebeneinander angezeigt, wobei die Unterschiede in den beiden Dateien grün markiert sind. Dateien können als XML-Dokumente (unter Berücksichtigung der Struktur und Semantik der Tags) oder als Textdokumente verglichen werden.

So vergleichen Sie eine geöffnete Datei mit einer anderen Datei:

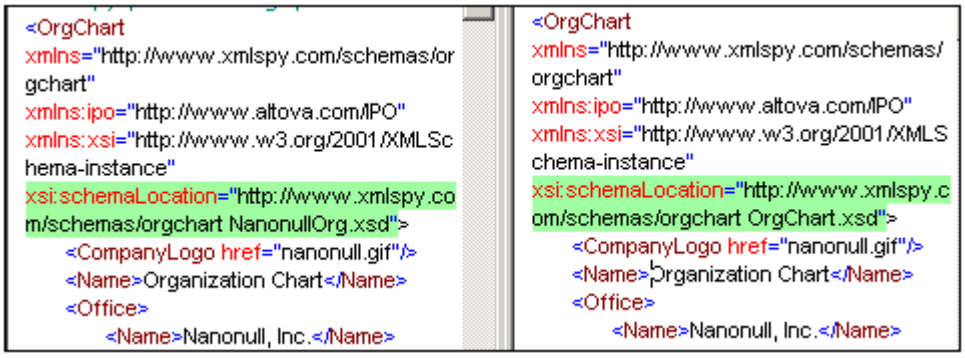
1. Wenn die Datei im Hauptfenster geöffnet ist (im Hauptfenster kann immer nur jeweils eine Datei aktiv sein), klicken Sie auf **Extras | Geöffnete Datei vergleichen mit...**. Daraufhin wird das folgende Dialogfeld angezeigt:

2. Suchen Sie die Datei, mit der Sie die geöffnete Datei vergleichen möchten, und wählen Sie sie aus. Wenn die gewünschte Datei im Hauptfenster bereits geöffnet (aber nicht aktiv) ist oder wenn sich die Datei in einem Projekt befindet, klicken Sie auf die Schaltfläche **Fenster** und wählen Sie die Datei aus.
3. Bestätigen Sie mit OK. Die zweite Datei wird geöffnet und das Dialogfeld "Einstellungen" wird angezeigt:



Diese Einstellungen werden unter Vergleichseinstellungen beschrieben. Wenn Sie nicht möchten, dass dieses Dialogfeld jedes Mal angezeigt wird, wenn Sie einen Dateivergleich starten, deaktivieren Sie das Kontrollkästchen "Einstellungen vor Vergleich anzeigen" am unteren Rand des Dialogfelds.

4. Wählen Sie die gewünschten Einstellungen aus und klicken Sie auf OK. Die beiden Dateien werden nun im Hauptfenster angezeigt. Zeilen, die Unterschiede aufweisen, erscheinen grün markiert. Der jeweils markierte Unterschied wird in beiden Dateien in einem dunkleren Grün angezeigt.



Außerdem wird das Steuerungsfenster "Dateien vergleichen" angezeigt:

 © 2005 Altova GmbH

Um den nächsten Unterschied auszuwählen, klicken Sie auf die Schaltfläche "**Weiter"**. Mit Hilfe der Schaltflächen **Erste**, **Letzte** und **Vorherig** können Sie von einem Unterschied zu anderen navigieren und einen Unterschied markieren.
Im Bereich "Unterschiede zusammenführen" können Sie die markierten unterschiedlichen Stellen von einer Datei in die andere kopieren. Um das Zusammenführen von Dateien zu aktivieren, müssen die folgenden Vergleichoptionen eingestellt sein: Die Option "Detaillierter Vergleich" muss aktiviert sein und die Option "Node-Tiefe ignorieren" muss deaktiviert sein. Wenn Sie einen Zusammenführvorgang rückgängig machen wollen, stoppen Sie den Vergleich, wählen Sie die Datei, in der die Änderung rückgängig gemacht werden soll, und klicken Sie auf den Befehl **Bearbeiten | Rückgängig** oder drücken Sie **Strg + Z**.

5. Nach Abschluss der Überprüfung und/oder des Zusammenführens von Unterschieden, klicken Sie auf "Fertig".

Hinweis:

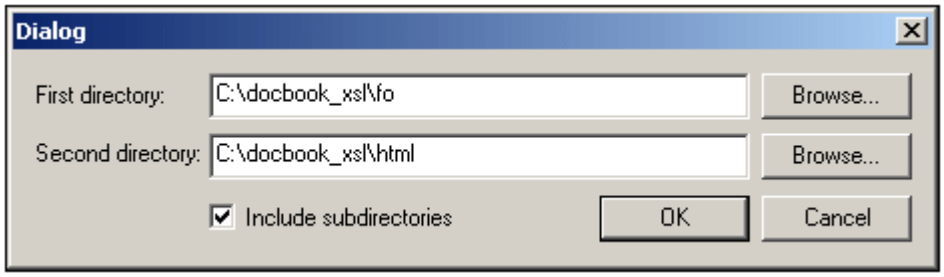
- Während des Vergleichsvorgangs können keine Bearbeitungen vorgenommen werden und auch die Ansicht kann nicht gewechselt werden. Bei Versuchen, die Ansicht einer der Dateien zu wechseln oder eine Datei zu bearbeiten, wird eine Meldung angezeigt mit der Warnung, dass dadurch der Vergleich beendet wird.
- Sie können während eines Vergleichs die Vergleichseinstellungen ändern (durch Auswahl von **Extras | Vergleichseinstellungen...**), doch werden die Änderungen erst beim nächsten Vergleichsvorgang wirksam; die neuen Einstellungen haben keinen Einfluss auf die aktuelle Sitzung.

Verzeichnisse vergleichen...

Mit dem Befehl "Verzeichnisse vergleichen..." können Sie zwei Verzeichnisse einschließlich oder ausschließlich ihrer Unterverzeichnisse vergleichen. Der Vergleich von Verzeichnissen dient dazu, festzustellen, welche Dateien fehlen und ob Dateien desselben Namens Unterschiede aufweisen oder nicht.

So vergleichen Sie zwei Verzeichnisse:

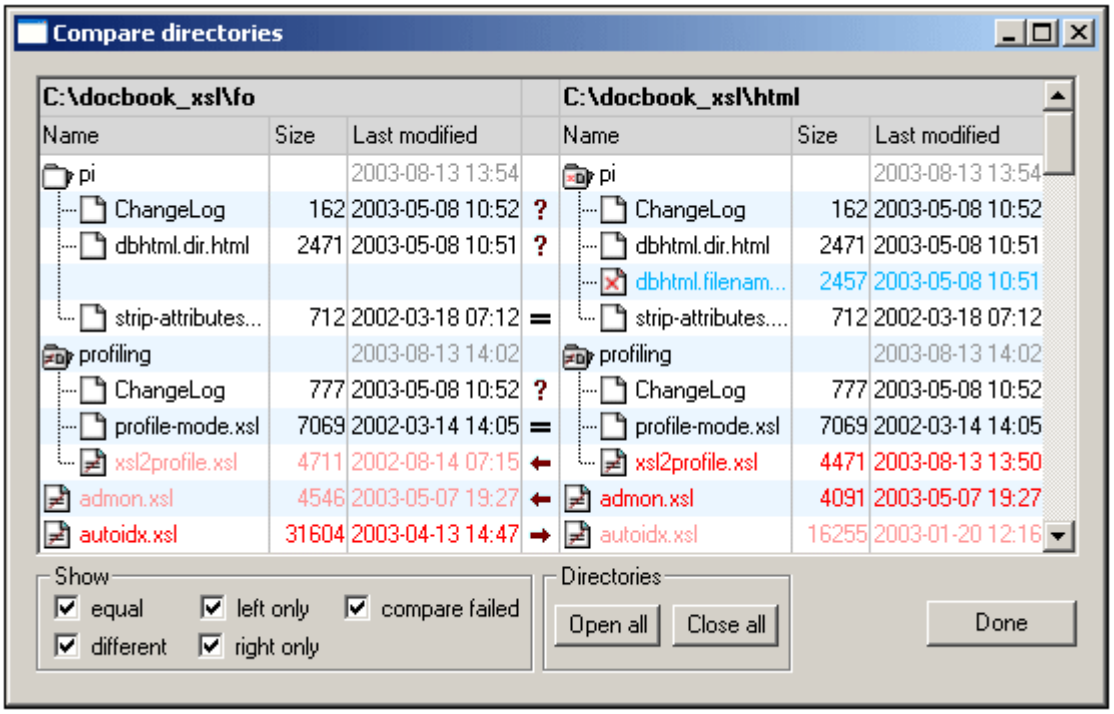
1. Klicken Sie auf **Extras | Verzeichnisse vergleichen**.Daraufhin wird das folgende Dialogfeld angezeigt.



2. Suchen Sie die zu vergleichenden Verzeichnisse und aktivieren Sie das Kontrollkästchen "Unterverzeichnisse berücksichtigen", wenn Unterverzeichnisse ebenfalls einbezogen werden sollen.

3. Klicken Sie auf OK. Das Dialogfeld "Einstellungen" wird angezeigt (Beschreibung siehe Vergleichseinstellungen).
4. Wählen Sie die gewünschten Einstellungen für den Dateivergleich aus.
5. Klicken Sie auf OK. Daraufhin erscheint ein Dialogfeld, in dem Sie den Verlauf des Vergleichs beobachten können.

Das Ergebnis des Vergleichs wird in einem Fenster angezeigt und sieht etwa folgendermaßen aus:



Verzeichnissymbole
Alle Verzeichnisnamen werden schwarz angezeigt.

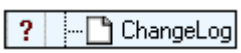
Dieses Verzeichnis enthält Dateien, die entweder alle nicht verglichen werden konnten oder sich nicht von der gleichnamigen Datei im anderen Verzeichnis unterscheiden.

Dieses Verzeichnis enthält eine oder mehrere Dateien, die im Verzeichnis, mit dem es verglichen wird, nicht vorhanden sind.

Dieses Verzeichnis enthält eine oder mehrere Dateien, die sich von der gleichnamigen Datei im anderen Verzeichnis unterscheiden.

Hinweis: Wenn ein Verzeichnis sowohl eine Datei enthält, die im verglichenen Verzeichnis nicht existiert, als auch eine Datei, die sich von der gleichnamigen Datei im verglichenen Verzeichnis unterscheidet, wird für das Verzeichnis das Symbol angezeigt.

Dateisymbole
Je nach Vergleichsstatus werden die Dateinamen in unterschiedlichen Farben angezeigt. Zur Kennzeichnung werden die folgenden Farben verwendet:



Diese Datei kann nicht (mit der gleichnamigen Datei im anderen Verzeichnis) verglichen werden. In der mittleren Spalte erscheint ein Fragezeichen. Der Dateiname wird schwarz angezeigt.

 © 2005 Altova GmbH

Diese Datei unterscheidet sich nicht von der gleichnamigen Datei im anderen Verzeichnis. In der mittleren Spalte wird ein Ist gleich-Zeichen angezeigt. Der Dateiname erscheint schwarz.

Diese Datei existiert im verglichenen Verzeichnis nicht. Die mittlere Spalte ist leer. Der Dateiname wird blau angezeigt.

Diese Datei weist Unterschiede zur gleichnamigen Datei im verglichenen Verzeichnis auf und die letzte Änderung dieser Datei ist jüngeren Datums als die letzte Änderung der verglichenen Datei. In der mittleren Spalte ist ein Pfeil zu sehen, der auf die jüngere Datei weist. Der Dateiname erscheint dunkelrot.

Diese Datei weist Unterschiede zur gleichnamigen Datei im verglichenen Verzeichnis auf und die letzte Änderung ist älteren Datums als die letzte Änderung der verglichenen Datei. In der mittleren Spalte wird ein Pfeil angezeigt, der in Richtung dieser (jüngeren) Datei weist. Der Dateiname erscheint hellrot.

Anzeigeoptionen

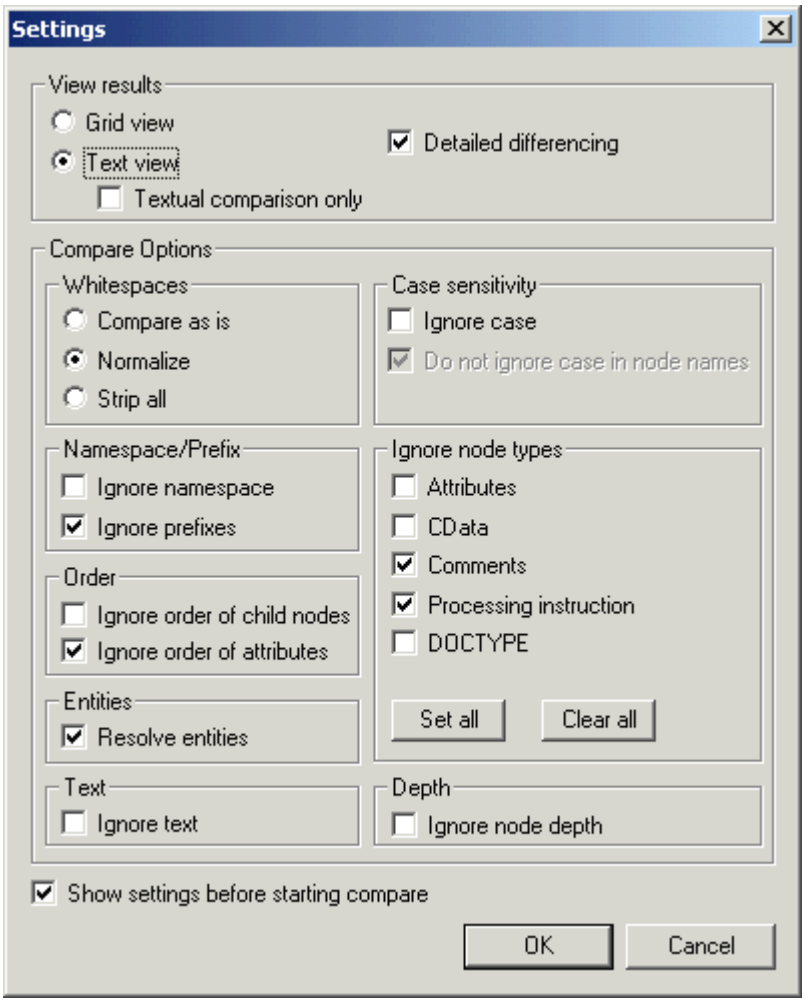
- Wählen Sie aus, welche Dateien angezeigt werden sollen. Aktivieren bzw. deaktivieren Sie dazu im Anzeigebereich am unteren Rand des Ergebnisfensters die entsprechenden Optionen.
- Öffnen oder schließen Sie alle Unterverzeichnisse durch Klicken auf die entsprechende Schaltfläche im Bereich "Verzeichnisse".
- Erweitern bzw. reduzieren Sie Unterverzeichnisse durch Doppelklicken auf das entsprechende Ordnersymbol.
- Die Spalten "Größe" und "Zuletzt geändert" können ein und ausgeblendet werden, indem Sie mit der rechten Maustaste auf eine Dateikopfzeile klicken und auf "Größe" und "Zuletzt geändert" klicken.

- Durch Ziehen der Spalten können Sie die Spaltenbreite ändern.
- Sie können das Ergebnisfenster maximieren, minimieren oder seine Größe anpassen.

Vergleichen und Zusammenführen von Dateien
Wenn Sie auf eine Zeile doppelklicken, werden beide in dieser Zeile befindlichen Dateien im Hauptfenster geöffnet, und an den beiden Dateien wird ein Dateivergleich durchgeführt. Sie können anschließend genau wie in einer normalen Vergleichssitzung fortfahren (siehe "Geöffnete Datei vergleichen mit").

Vergleichseinstellungen

Wenn Sie auf **Vergleichseinstellungen** klicken, wird das unten gezeigte Dialogfeld "Einstellungen" angezeigt, in dem Sie die Einstellungen für Ihre Vergleichssitzungen festlegen. Die Einstellungen, die beim Start einer Vergleichssitzung aktiv sind, sind diejenigen, die auf diese Sitzung angewendet werden.



Ergebnisse anzeigen
Wählt die Ansicht aus, in der die Ergebnisse angezeigt werden sollen. Sie haben die Wahl zwischen folgenden Möglichkeiten:

- Grid-Ansicht (XML-Vergleich)
- Textansicht; die Option "Nur Textvergleich" ist deaktiviert (XML-Vergleich)
- Textansicht; die Option "Nur Textvergleich" ist aktiviert (Textvergleich)

Wenn Sie eine Ansicht auswählen, die den XML-Vergleich unterstützt, werden die Dokumente als XML-Dokumente behandelt und die XML-Vergleichsoptionen werden aktiviert. Wird Textvergleich ausgewählt, werden nur diejenigen Optionen aktiviert, die den Textvergleich betreffen (Whitespaces und Groß- und Kleinschreibung); alle anderen Vergleichsoptionen sind deaktiviert.

Hinweis: Sie können Dateien sowohl in der Grid-Ansicht als auch in der Textansicht und sowohl im XML- als auch im Textvergleichsmodus zusammenführen. Wenn Sie eine Zusammenführung rückgängig machen möchten, stoppen sie den Vergleich, wählen Sie die Datei aus, in der die Änderung rückgängig gemacht werden soll und klicken Sie auf **Bearbeiten | Rückgängig** oder drücken Sie **Strg + Z**.

Detaillierter Vergleich
Ist diese Option deaktiviert, werden Unterschiede zwischen unmittelbaren Geschwisterelementen als ein einziger Unterschied dargestellt und die Zusammenführung wird

 © 2005 Altova GmbH

deaktiviert. Ist die Option aktiv, werden Unterschiede zwischen unmittelbaren Geschwisterelementen als separate Unterschiede markiert und die Zusammenführung wird aktiviert.

Hinweis: Um die Zusammenführung von Dateien zu aktivieren, muss die Option "Detaillierter Vergleich" aktiviert sein.

Whitespaces
Bei Leerzeichen handelt es sich um Leerzeichen, Tabulatorzeichen, Wagenrücklauf und Zeilenvorschub. Die drei Optionen in diesem Bereich dienen zum Vergleichen von Dateien, bei denen die Leerzeichen identisch sind, zum Vergleich von Dateien bei denen die Leerzeichen vereinheitlicht wurden (d.h. alle aufeinander folgenden Leerzeichen werden als ein Leerzeichen betrachtet) und zum Vergleich von Dateien, bei denen alle Leerzeichen entfernt wurden. Die Option "Whitespaces" steht sowohl für XML- als auch für Textvergleiche zur Verfügung.

Groß-/Kleinschreibung
Ist das Kontrollkästchen "Groß-/Kleinschreibung ignorieren" aktiviert, haben Sie die Option, die Groß- und Kleinschreibung in Node-Namen zu ignorieren oder zu berücksichtigen (nur bei XML-Vergleichen). Die Option "Groß-/Kleinschreibung" steht sowohl für XML- als auch für Textvergleiche zur Verfügung.

Namespace/Präfix
Diese Optionen dienen zum Ignorieren von Namespaces und Präfixen beim Suchen nach Unterschieden.

Reihenfolge
Wenn die Option "Child-Node Reihenfolge ignorieren" ausgewählt ist, spielt die Position der Child Nodes relativ zueinander keine Rolle. Der Vergleich wird an der Gesamtheit aller Child Nodes durchgeführt und wenn der einzige Unterschied zwischen einem Child Node in dem einen Dokument und einem Child Node in dem damit verglichenen Dokument die relative Position im Nodeset ist, so wird dieser Unterschied ignoriert. Beachten Sie jedoch, dass jeder Child-Element-Node an seinem Namen, seinen Attributen und seiner Position identifiziert wird. Haben mehrere Nodes im Geschwister-Nodeset denselben Namen, so werden diese Nodes anhand ihrer Position identifiziert, selbst wenn die Option "Child-Node Reihenfolge ignorieren" ausgewählt ist. Diese Option wird auf jede Ebene separat angewendet.

Ist die Option "Child-Node Reihenfolge ignorieren" deaktiviert, so werden die Unterschiede in der Reihenfolge der Nodes als Unterschiede markiert.

Des Weiteren steht eine Option zum Ignorieren der Reihenfolge der Attribute zur Verfügung. Sie wird auf die Reihenfolge der Attribute eines einzelnen Elements angewendet.

Entities
Ist die Option "Entities auflösen" ausgewählt, werden alle Entitäten im Dokument aufgelöst. Andernfalls werden die Dateien mit den Entitäten in ihrem gegenwärtigen Zustand verglichen.

Text
Wenn Sie die Option "Text ignorieren" auswählen, werden die Unterschiede in einander entsprechenden Text-Nodes nicht berücksichtigt.

Node-Typen ignorieren
Aktivieren Sie das Kontrollkästchen für die Node-Typen, die beim Dateivergleich nicht berücksichtigt werden sollen. Zur Wahl stehen die Node-Typen Attribute, CDATA, Comments, Processing Instructions und DOCTYPE statements.

Tiefe
Ist die Option "Node-Tiefe ignorieren" ausgewählt, so wird die Tiefe eines Elements (d.h. weitere

Ebenen mit Nachfahren dieses Elements) relativ zur Tiefe des entsprechenden Elements in der verglichenen Datei ignoriert. Um die Zusammenführung zu ermöglichen, muss diese Option deaktiviert werden.

Einstellungen vor Vergleich anzeigen
Wenn Sie diese Option aktivieren, wird das Dialogfeld "Einstellungen" jedes Mal angezeigt, wenn Sie auf den Befehl "**Geöffnete Datei vergleichen mit"** klicken. Auf diese Art können Sie die Einstellungen vor jedem Vergleich überprüfen und ändern.

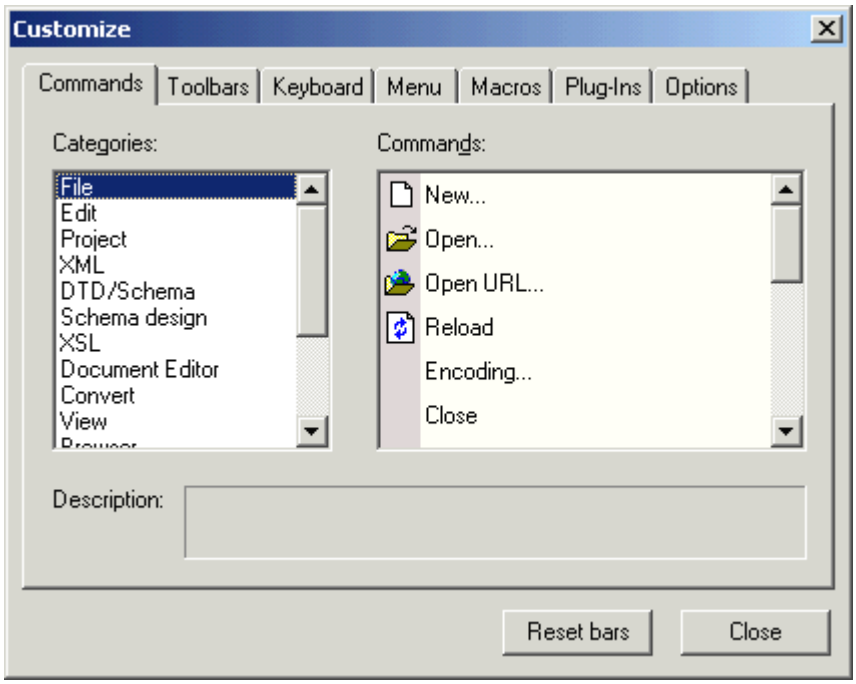
Ist dieses Kontrollkästchen deaktiviert, wird der Vergleich direkt gestartet, wenn Sie den entsprechenden Befehl wählen.

13.14.7 Anpassen...

Mit diesem Befehl können Sie XMLSpy 2005 an Ihre eigenen Bedürfnisse anpassen.

Befehle

Auf dem Register Befehle können Sie Ihre Menüs oder Symbolleisten anpassen.



So fügen Sie einen Befehl zu einer Symbolleiste oder einem Menü hinzu:

1. Öffnen Sie das Dialogfeld über die Menübefehle **Extras | Anpassen.**
2. Wählen Sie die Befehlskategorie im Kategorie-Listenfeld. Die verfügbaren Befehle werden im entsprechenden Listenfeld angezeigt.
3. Markieren Sie einen Befehl aus dem Listenfeld, und ziehen Sie ihn in ein existierendes Menü oder eine Symbolleiste.
4. Ein **I**-Zeichen wird angezeigt, wenn Sie den Cursor an eine Stelle bewegen, an die der ausgewählte Befehl gestellt werden kann.
5. Lassen Sie die Maustaste los, um den Befehl an der gewünschten Stelle einzufügen.

- An der Spitze des Mauszeigers wird beim Ziehen eines Befehls eine kleine Schaltfläche

 © 2005 Altova GmbH

angezeigt. Das Zeichen unterhalb des Zeigers zeigt an, dass der Befehl nicht an diese Position gestellt werden kann.
- Das Zeichen verschwindet, sobald Sie eine Position erreichen, an die der Befehl gestellt werden kann (innerhalb eines Menüs oder einer Symbolleiste).
- Wenn Sie den Mauszeiger über ein Menü halten, wird dieses geöffnet, so dass Sie den Befehl an einer beliebigen Stelle im Menü einfügen können.
- Befehle können in Menüs oder Symbolleisten eingefügt werden. Wenn Sie eine eigene Symbolleiste erzeugt haben, können Sie nun die gewünschten Befehle einfügen.

Bitte beachten Sie:
Sie können mit dieser Methode auch die Befehle der **Kontextmenüs** bearbeiten (rechtsklicken Sie auf eine beliebige Stelle im Fenster, und das Kontextmenü wird geöffnet). Klicken Sie auf das Register Menü und wählen Sie aus dem Dropdown-Listenfeld das gewünschte Kontextmenü aus.

So löschen Sie einen Befehl oder ein Menü:
1. Öffnen Sie dieses Dialogfeld über die Menübefehle **Extras | Anpassen.**
2. Klicken Sie auf den Menüeintrag oder das Symbol, welches Sie löschen möchten, und ziehen Sie es mit der Maus an eine Stelle, an der die kleine Schaltfläche am Zeiger angezeigt wird.
3. Lassen Sie die Maustaste los, und der Befehl wird entfernt.

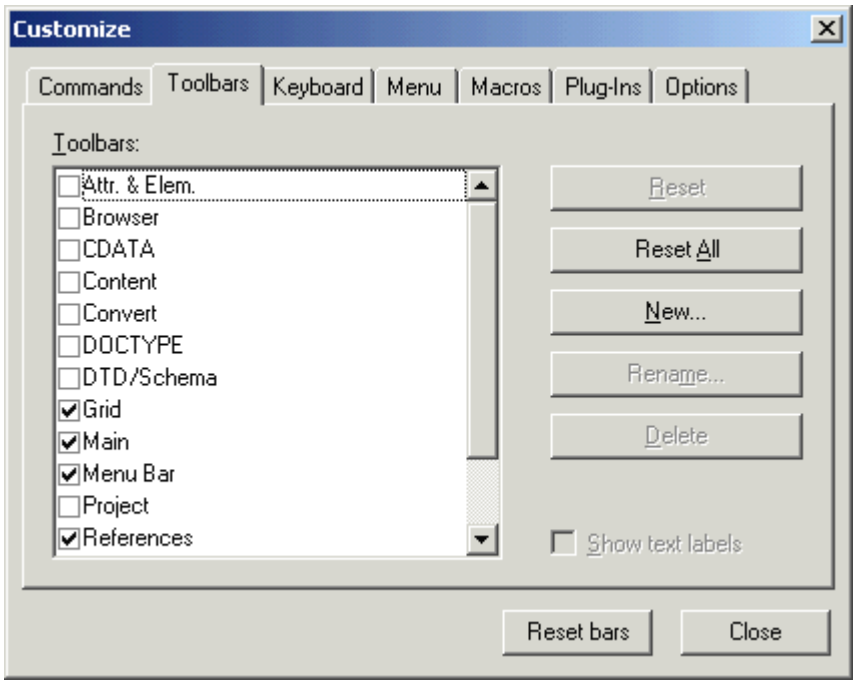
Symbolleisten

Auf dem Register Symbolleisten können Sie spezifische Symbolleisten aktivieren oder deaktivieren. Weiters können Sie eigene Symbolleisten erstellen.

XMLSpy 2005 Symbolleisten enthalten Symbole für die meistverwendeten Befehle.
Zu jedem Symbol gehört ein kurzer Tooltip, der angezeigt wird, wenn Sie mit dem Cursor auf das Symbol gehen. In der Statusleiste wird eine detailliertere Beschreibung zum jeweiligen Symbol angezeigt.

Sie können die Symbolleisten aus Ihrer Standardposition an eine andere beliebige Stelle am Bildschirm verschieben, sowie diese an den rechten oder linken Rand des Hauptfensters andocken.

- Symbolleisteneinstellungen, die in der Grid-, Schema/WSDL Design- oder Text-Ansicht definiert wurden, stehen in den jeweiligen Ansichten zur Verfügung. Die Symbolleisten der Browser-Ansicht sind unabhängig von der jeweiligen Ansicht immer verfügbar.



So aktivieren oder deaktivieren Sie eine Symbolleiste:

1. Klicken Sie auf das Kontrollkästchen um die jeweilige Symbolleiste zu aktivieren oder deaktivieren.

So erstellen Sie eine neue Symbolleiste:

1. Klicken Sie auf die Schaltfläche **Neu...** und geben Sie der neuen Symbolleiste einen Namen (Dialogfeld Symbolleistenname).
2. Fügen Sie Befehle in die neue Symbolleiste ein (Register Befehle im Dialogfeld **Anpassen**).

So setzen Sie die Menüleiste zurück

- Klicken Sie auf den Menüleisteneintrag und
- Klicken Sie auf die Schaltfläche **Zurücksetzen**, um die Menübefehle auf Werkseinstellung zurückzusetzen.

So setzen Sie alle Symbolleisten und Menübefehle zurück

- Klicken Sie auf die Schaltfläche **Alles Zurücksetzen**, um die Befehle der Symbolleisten auf Werkseinstellung zurückzusetzen. Sie erhalten eine Meldung, dass nun alle Symbolleisten und Menüs zurückgesetzt werden.
- Klicken Sie auf Ja um das Zurücksetzen zu bestätigen.

So ändern Sie den Namen einer Symbolleiste:

- Klicken Sie auf die Schaltfläche **Umbenennen...** um den Namen der Symbolleiste zu ändern.

So löschen Sie eine Symbolleiste:

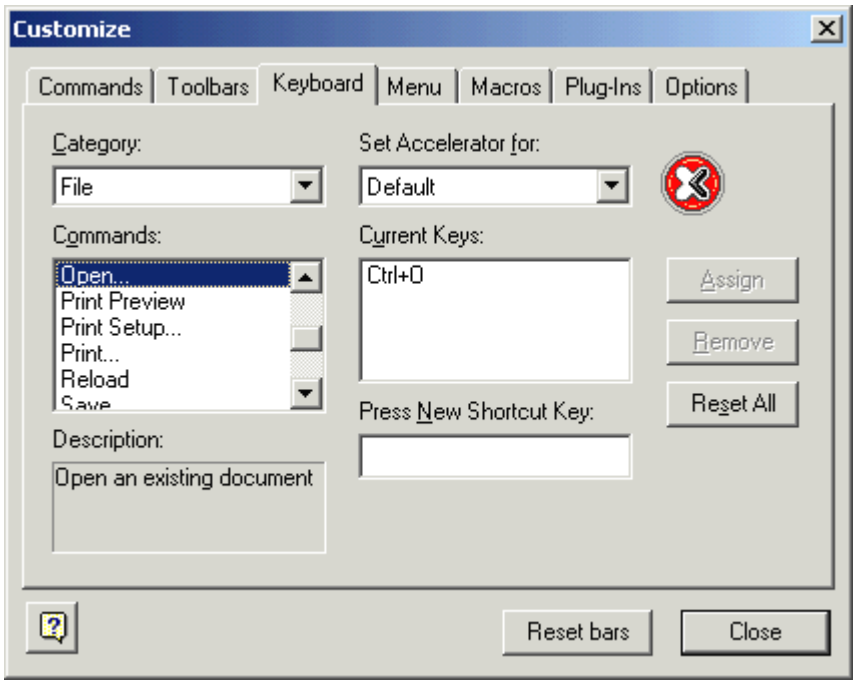
- Klicken Sie auf die Schaltfläche **Löschen** um die im Listenfeld markierte Symbolleiste zu löschen.
- Sie erhalten eine Meldung, in der Sie gefragt werden, ob die markierte Symbolleiste tatsächlich gelöscht werden soll. Bestätigen Sie den Löschvorgang mit Ja.
- oder, rechtsklicken Sie auf die Menü- oder Symbolleiste und wählen Sie Löschen aus dem Popup-Menü.

© 2005 Altova GmbH

Textlabels zeigen:
Wenn Sie diese Option aktivieren, werden Hilfetexte zu den Symbolen angezeigt.

Tastatur

Auf dem Register Tastatur können Sie Tastaturkürzel für jeden Befehl in XMLSpy 2005 definieren (oder löschen).



So weisen Sie einem Befehl ein neues Kürzel zu:

1. Wählen Sie die Befehlskategorie im Dropdown-Listenfeld **Kategorie**.
2. Wählen Sie den **Befehl** aus dem Listenfeld, welchem Sie ein neues Kürzel zuweisen möchten.
3. Klicken Sie in das "Tastaturkürzel eingeben:" Textfeld und geben Sie das Kürzel ein. Die Kürzel werden sofort im Textfeld angezeigt. Wenn das Kürzel schon einem anderen Befehl zugeordnet war, wird diese Funktion im Textfeld angezeigt.
4. Klicken Sie auf die Schaltfläche **Zuweisen**.
5. Das neue Kürzel wird in die Liste der aktuellen Kürzel aufgenommen.
   (um das Textfeld zu **löschen** drücken Sie die Strg-, ALT- oder Umschalttaste).

So heben Sie eine Zuweisung auf (oder löschen ein Kürzel):

1. Klicken Sie auf den Kürzel in der Liste der aktuellen Kürzel, der gelöscht werden soll und
2. Klicken Sie auf die Schaltfläche **Entfernen**, die jetzt aktiv ist.
3. Klicken Sie auf die Schaltfläche Schließen um die Änderungen zu speichern.

Kürzel setzen für:
Derzeit noch keine Funktion.

Zugewiesene Tastatur-Kürzel:

Kürzel nach Tasten	
F1	Hilfe-Menü
F3	Weitersuchen
F5	Aktualisieren
F7	Wohlgeformtheit prüfen
F8	Validieren
F9	Breakpoint einfügen/löschen
Strg+F9	Breakpoint ein/aus
F10	XSL Transformation
Strg+F10	XSL:FO Transformation
F11	Einsteigen
Strg+F11	Überspringen
Shift + F11	Aussteigen
Alt+F11	Debugger starten
Num +	Erweitern
Num -	Reduzieren
Num *	Alles erweitern
Strg+Num-	Unselektierte reduzieren
Strg + G	Gehe zu Zeichen/Zeile
Strg+TAB	und
Strg+F6	Offene Fenster durchblättern
Pfeiltasten oben/unten	Laufleiste auf/ab
Esc.	Dialogfeld schließen/Editieren abbrechen
Eingabetaste/Leerzeichentaste	Bestätigen
Alt + F4	Schließt XMLSpy 2005
Strg + F4	Schließt aktives Fenster
Alt + F, 1	Öffnet letzte Datei

© 2005 Altova GmbH

Strg + Doppelklick Element (Schema-Ansicht)	Elementdefinition anzeigen
Strg + N	Datei Neu
Strg + O	Datei Öffnen
Strg + S	Datei Speichern
Strg + P	Datei Drucken
Strg + A	Alles auswählen
Shift + Del	Ausschneiden (oder Strg + X)
Strg + C	Kopieren
Strg + V	Einfügen
Strg + Z	Rückgängig
Strg + Y	Wiederherstellen
Del	Löschen (Objekt in der Schema/Enhanced Grid Ansicht)
Strg + F	Suchen
F3	Weitersuchen
Strg + H	Ersetzen
Strg + I	Attribut anhängen
Strg + E	Element anhängen
Strg + T	Text anhängen
Strg + D	CDATA anhängen
Strg + M	Comment anhängen
Strg + SHIFT + I	Attribut einfügen
Strg + SHIFT + E	Element einfügen
Strg + SHIFT + T	Text content einfügen
Strg + SHIFT + D	CDATA einfügen
Strg + SHIFT + M	Comment einfügen
Strg + ALT + I	Child Attribute hinzu
Strg + ALT + E	Child Element hinzu
Strg + ALT + T	Child Text hinzu
Strg + ALT + D	Child CDATA hinzu
Strg + ALT + M	Child Comment hinzu

Kürzel nach Funktion

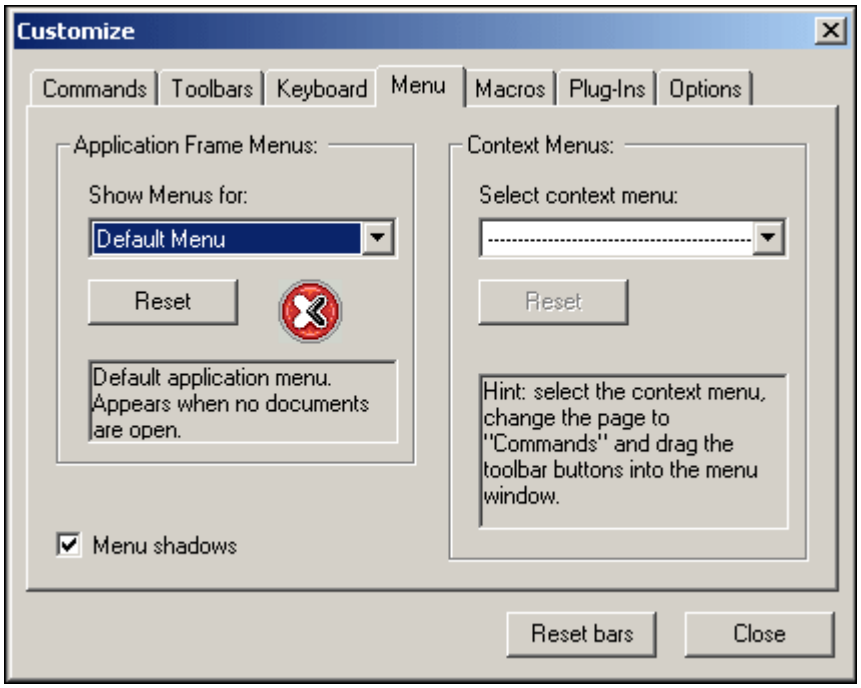
Editieren abbrechen	Esc.
Child Attribute hinzu	Strg + ALT + I
Child CDATA hinzu	Strg + ALT + D
Child Comment hinzu	Strg + ALT + M
Child Element hinzu	Strg + ALT + E
Child Text hinzu	Strg + ALT + T
Attribut anhängen	Strg + I
CDATA anhängen	Strg + D
Comment anhängen	Strg + M
Element anhängen	Strg + E
Text anhängen	Strg + T
Wohlgeformtheit prüfen	F7
Aktives Fenster schließen	Strg + F4
Schließt XMLSpy 2005	Alt + F4
Reduzieren	Num -
Unselektierte reduzieren	Strg + Num-
Bestätigen	Return / Space bar
Kopieren	Strg + C
Ausschneiden	SHIFT + Del (or Strg + X)
Offenen Fenster durchblättern	Strg + TAB und Strg + F6
Löschen	Del
Breakpoint aktivieren/deaktivieren	Strg + F9
Erweitern	Num +
Alles erweitern	Num *
Datei Neu	Strg + N
Datei Öffnen	Strg + O
Datei Drucken	Strg + P
Datei Speichern	Strg + S

© 2005 Altova GmbH

Suchen	Strg + F
Weitersuchen	F3
Gehe zu Zeichen/Zeile	Strg + G
Hilfemenü	F1
Attribut einfügen	Strg + SHIFT + I
CDATA einfügen	Strg + SHIFT + D
Comment einfügen	Strg + SHIFT + M
Element einfügen	Strg + SHIFT + E
Breakpoint einfügen/löschen	F9
Text content einfügen	Strg + SHIFT + T
Laufleiste auf/ab	Pfeiltasten oben/unten
Letzte Datei öffnen	Alt + F, 1
Einfügen	Strg + V
Wiederherstellen	Strg + Y
Aktualisieren	F5
Ersetzen	Strg + H
Alles markieren	Strg + A
Start Debugger	Alt + F11
Einsteigen	F11
Aussteigen	Shift + F11
Überspringen	Strg + F11
Definition anzeigen	Strg + Doppelklick Element
Rückgängig	Strg + Z
Validieren	F8
XSL Transformation	F10
XSL:FO Transformation	Strg + F10

Menü

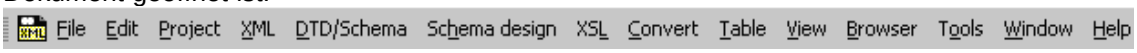
Auf dem Register Menü können Sie die Menüleisten und die Rechtsklick-Kontextmenüs anpassen.



Sie können sowohl die standardmäßigen Menüleisten, als auch die XMLSpy 2005 Menüleisten anpassen.

Das **Standardmenü** wird angezeigt, wenn Sie kein XML-Dokument in XMLSpy 2005 geöffnet haben.

Das XMLSpy 2005-Menü ist die Menüleiste, die dargestellt wird, wenn mindestens ein XML-Dokument geöffnet ist.



So passen Sie ein Menü an:

1. Wählen Sie die Menüleiste aus dem Dropdown-Listenfeld "Menüs anzeigen für:", welche Sie anpassen wollen.
2. Klicken Sie auf das Register **Befehle** und ziehen Sie die gewünschten Befehle in das ausgewählte Menü.

So löschen Sie Befehle aus einem Menü:

1. Rechtsklicken Sie bei geöffnetem Dialogfeld "Anpassen" auf den Befehl, oder das Symbol, welches für den Befehl steht, und
2. Wählen Sie die Option Löschen aus dem Popup Menü,
   oder
1. Wählen Sie **Extras | Anpassen** um das Dialogfeld Anpassen zu öffnen und
2. Ziehen Sie den Befehl aus dem Menü bis am Mauszeiger die kleine Schaltfläche angezeigt wird.

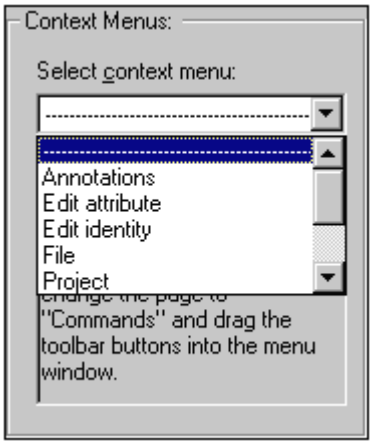
So setzen Sie eine der Menüleisten zurück:

1. Wählen Sie entweder den Eintrag Standard oder XMLSpy 2005 aus dem Dropdown-Listenfeld und
2. Klicken Sie auf die Schaltfläche **Zurücksetzen** unter dem Menünamen.
   Mit einer Meldung werden Sie gefragt, ob Sie die Menüleiste tatsächlich zurücksetzen möchten.

© 2005 Altova GmbH

So passen Sie ein Kontextmenü an (Rechtsklick-Menü):

1. Wählen Sie das entsprechende Kontextmenü aus dem Dropdown-Listenfeld.
2. Klicken Sie auf das Register **Befehle** und ziehen Sie die Befehle in das offene Kontextmenü.



So löschen Sie Befehle aus einem Kontextmenü:

1. Rechtsklicken Sie bei geöffnetem Dialogfeld "Anpassen" auf den Befehl, oder das Symbol, welches für den Befehl steht, und
2. Wählen Sie die Option Löschen aus dem Popup Menü,

oder

1. Wählen Sie **Extras | Anpassen** um das Dialogfeld Anpassen zu öffnen und
2. Ziehen Sie den Befehl aus dem Menü heraus und lassen Sie die Maustaste los, sobald am Mauszeiger die kleine Schaltfläche angezeigt wird.

So setzen Sie eines der Kontextmenüs zurück:

1. Wählen Sie das Kontextmenü aus dem Dropdown-Listenfeld und
2. Klicken Sie auf die Schaltfläche **Zurücksetzen** unter dem Menünamen. Mit einer Meldung werden Sie gefragt, ob Sie das Kontextmenü tatsächlich zurücksetzen möchten.

So schließen Sie ein Kontextmenüfenster:

1. Klicken Sie auf das Symbol **Schließen** rechts oben in der Titelleiste oder
2. Klicken Sie auf die Schaltfläche "Schließen" im Dialogfeld "Anpassen".

Menüanimationen (vor Windows 2000)

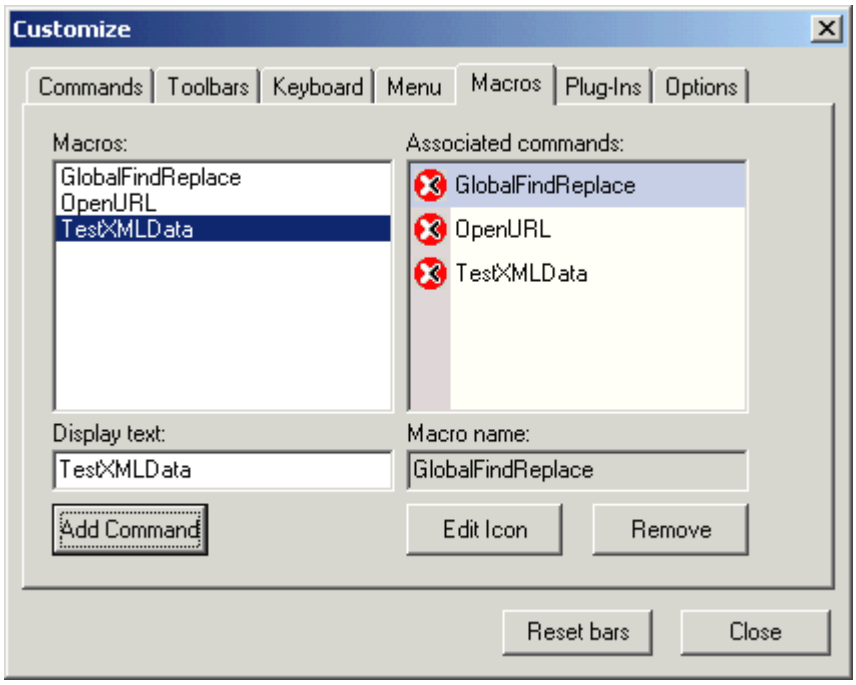
- Wählen Sie eine der Menüanimationen aus dem Dropdown-Listenfeld, wenn Sie animierte Menüs sehen möchten. Bitte beachten Sie, dass das Dropdown-Listenfeld nur für Windows-Versionen vor Windows 2000 verfügbar ist. Ab Windows 2000 können die Einstellungen auf dem Register Effekte im Dialogfeld "Eigenschaften von Anzeige" geändert werden. Doppelklicken Sie in der Systemsteuerung auf das Symbol **Anzeigen**, um das Dialogfeld zu öffnen.

Menüschatten

- Klicken Sie auf das Kontrollkästchen Menüschatten, wenn Menüs mit Schatten dargestellt werden sollen.

Makros

Auf dem Register Makros können Sie Makros, die Sie zuvor in der XML Skript-Umgebung erzeugt wurden, in eine Symbolleiste oder ein Menü einfügen.



So fügen Sie das Makrosymbol in eine Symbolleiste oder ein Menü ein:

1. Starten Sie die Skriptumgebung über **Extras | Skripting-Umgebung**.
2. Doppelklicken Sie auf den Eintrag XMLSpyMacros im Ordner Module des Projekts **XMLSpyGlobalScripts**.
   Zuvor definierte Makros werden im rechten Fenster angezeigt.
3. Wechseln Sie zurück in XMLSpy 2005 und wählen Sie **Extras | Anpassen** und klicken Sie auf das Register Makros.
   Die in der Skriptumgebung definierten Makros werden nun im Makros Listenfeld links angezeigt.
4. Klicken Sie auf den Namen des Makros und dann die Schaltfläche "Befehl hinzufügen". Der Makroname wird im Listenfeld "Zugewiesene Befehle" angezeigt.
5. Klicken Sie im Listenfeld "Zugewiesene Befehle" auf den Namen des Makros und ziehen Sie ihn in die Symbolleiste oder das Menü.

- Zum Bearbeiten des Symbols für ein Makro, klicken Sie auf die Schaltfläche **Symbol bearbeiten**.

- Um ein Makro aus dem Listenfeld "Zugewiesene Befehle" zu löschen, klicken Sie auf die Schaltfläche Entfernen.

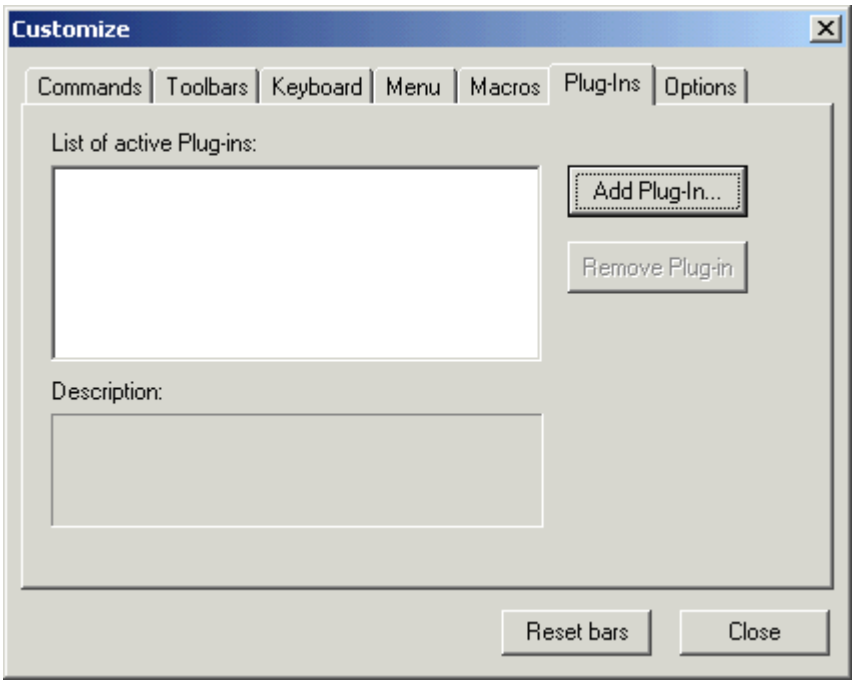
Plug-Ins

Auf dem Register Plug-Ins können Sie Plug-Ins in eine Symbolleiste oder ein Menü einfügen.

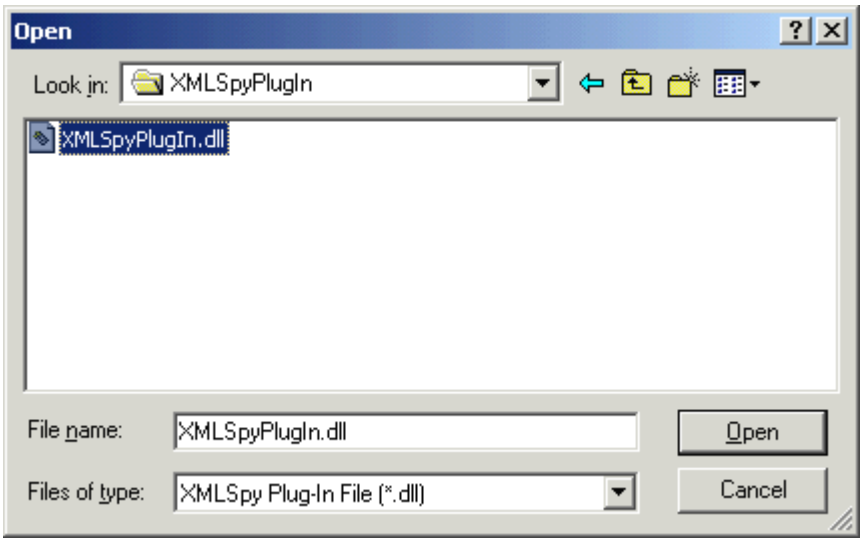
So fügen Sie das Symbol für ein Plug-In in eine Symbolleiste oder Menü ein:

1. Klicken Sie auf die Schaltfläche "**Plug-In hinzufügen**".

© 2005 Altova GmbH



2. Wählen Sie den Ordner aus und klicken Sie auf die Plug-In Datei um diese zu markieren (in diesem Beispiel: XMLSpyPlugIn.dll).



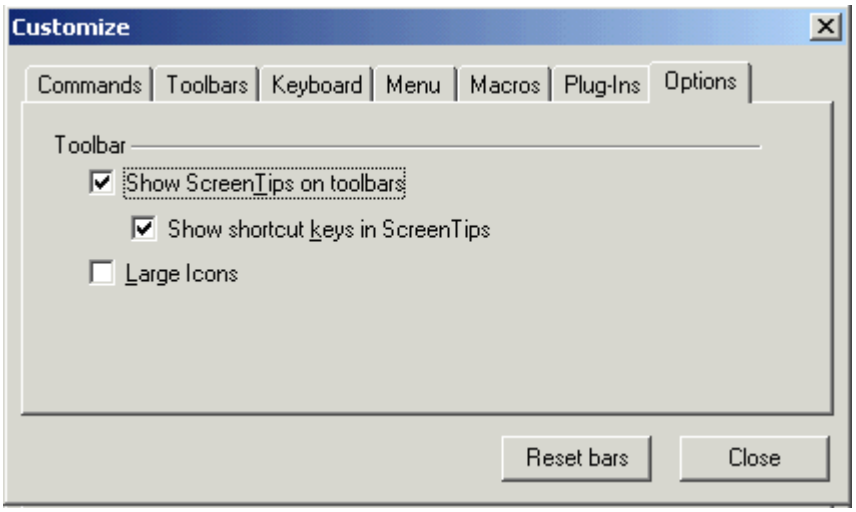
3. Klicken Sie auf die Schaltfläche **Öffnen** um das Plug-In zu installieren.
   Der Name des Plug-Ins wird in der Liste "Derzeit aktive Plug-Ins" angezeigt und das Symbol des/der Plug-Ins wird in einer neuen Symbolleiste angezeigt.

So entfernen Sie ein Plug-In:

1. Klicken Sie auf den Namen des Plug-Ins in der Liste "Derzeit aktive Plug-Ins" und klicken Sie auf die Schaltfläche "Plug-In löschen".

Optionen

Auf dem Register Optionen können Sie verschiedene, allgemeine Einstellungen vornehmen.



Symbolleiste

Wenn das Kontrollkästchen **Tooltip in Symbolleiste einblenden** aktiviert ist, wird ein Popup beim Ziehen des Mauszeigers über das entsprechende Symbol in der Symbolleiste angezeigt. Das Popup enthält eine Kurzbeschreibung der Funktion, sowie, falls vorhanden, ein Tastaturkürzel.

Das Kontrollkästchen "Tastaturkürzel in Tooltips anzeigen" ermöglicht es Ihnen, auszuwählen, ob die Tastaturkürzel (wo vorhanden) angezeigt werden sollen.

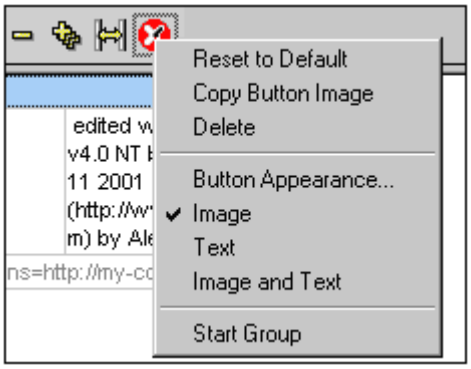
Mit dem Kontrollkästchen Große Symbole können Sie die Größe der angezeigten Symbole beeinflussen.

Kontextmenü 'Anpassen'

Über das Kontextmenü "Anpassen" können Sie Symbole und Menüobjekte Ihren Ansprüchen entsprechend anpassen.

So öffnen Sie das Kontextmenü "Anpassen":

1. Zuerst öffnen Sie das Menü Anpassen über den Menübefehl **Extras | Anpassen**.
2. Dann setzen Sie den Mauszeiger auf das Symbol (oder Menü) und **rechtsklicken Sie**, um das Kontextmenü zu öffnen.



© 2005 Altova GmbH

Zurücksetzen
Derzeit keine Funktion.

Schaltflächensymbol kopieren
Das Symbol wird in die Zwischenablage kopiert.

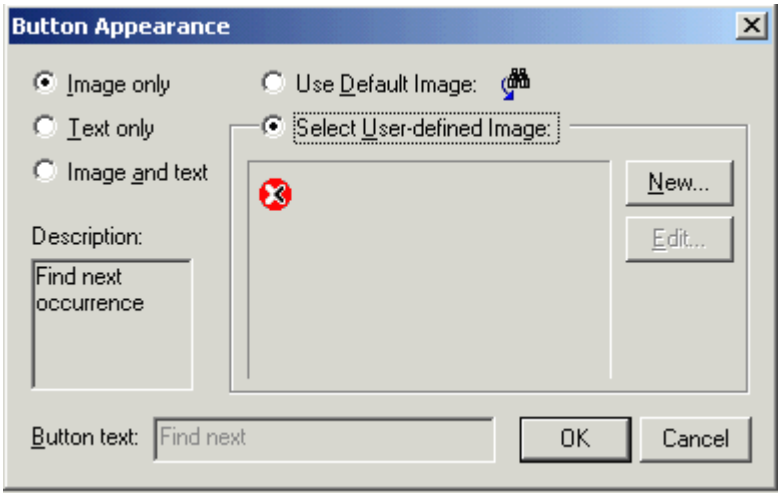
Löschen
Mit dieser Option wird das Symbol oder Menü gelöscht, auf welches Sie zuvor rechtsgeklickt haben. Wenn Sie ein Menü löschen, werden alle enthaltenen Menüoptionen gelöscht!

So stellen Sie ein gelöschtes Menü wieder her:

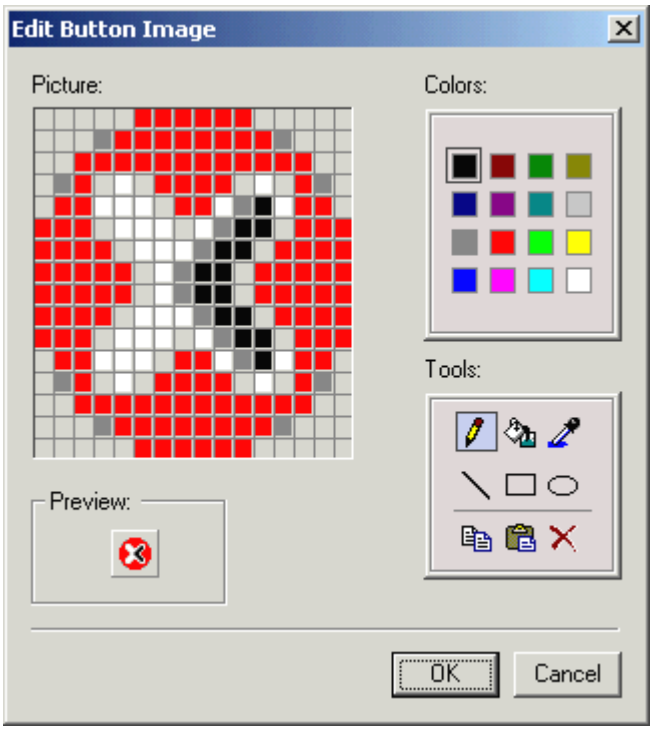
- Wählen Sie das Register Menü
- Wählen Sie das Menü, welches wiederhergestellt werden soll (XMLSpy 2005 oder Standardmenü).
- Klicken Sie auf die Schaltfläche Zurücksetzen unter dem Menü im Dropdown-Listenfeld.

Schaltflächendarstellung
Mit dieser Option können Sie Schaltflächensymbole und Text bearbeiten. Derzeit kann nur das Symbol für Makros bearbeitet werden (Standardsymbol XMLSpy 2005).

Mit den Optionsfeldern "Nur Symbol", "Nur Text" oder "Symbol und Text" können Sie definieren, was Sie editieren möchten. Klicken Sie auf das Optionsfeld **Benutzerdefiniertes Symbol** und klicken Sie auf eines der verfügbaren Symbole.



Klicken Sie auf die Schaltfläche "Bearbeiten" um das Dialogfeld **Schaltflächensymbol editieren** zu öffnen. Der Schaltflächentext kann bearbeitet werden, wenn Sie entweder die Option "Nur Text" oder "Symbol und Text" ausgewählt haben.



Symbol
Mit dieser Option ändern Sie das graphische Design des Symbols.

Text
Mit dieser Option ändern Sie die Textbeschreibung.

Symbol und Text
Mit dieser Option wird die gleichzeitige Darstellung des Symbols und des Textes aktiviert.

Gruppe starten
Mit dieser Option wird links neben dem Symbol, auf welches Sie rechtsgeklickt haben, ein vertikaler Trennungsstrich angezeigt.

13.14.8 Optionen

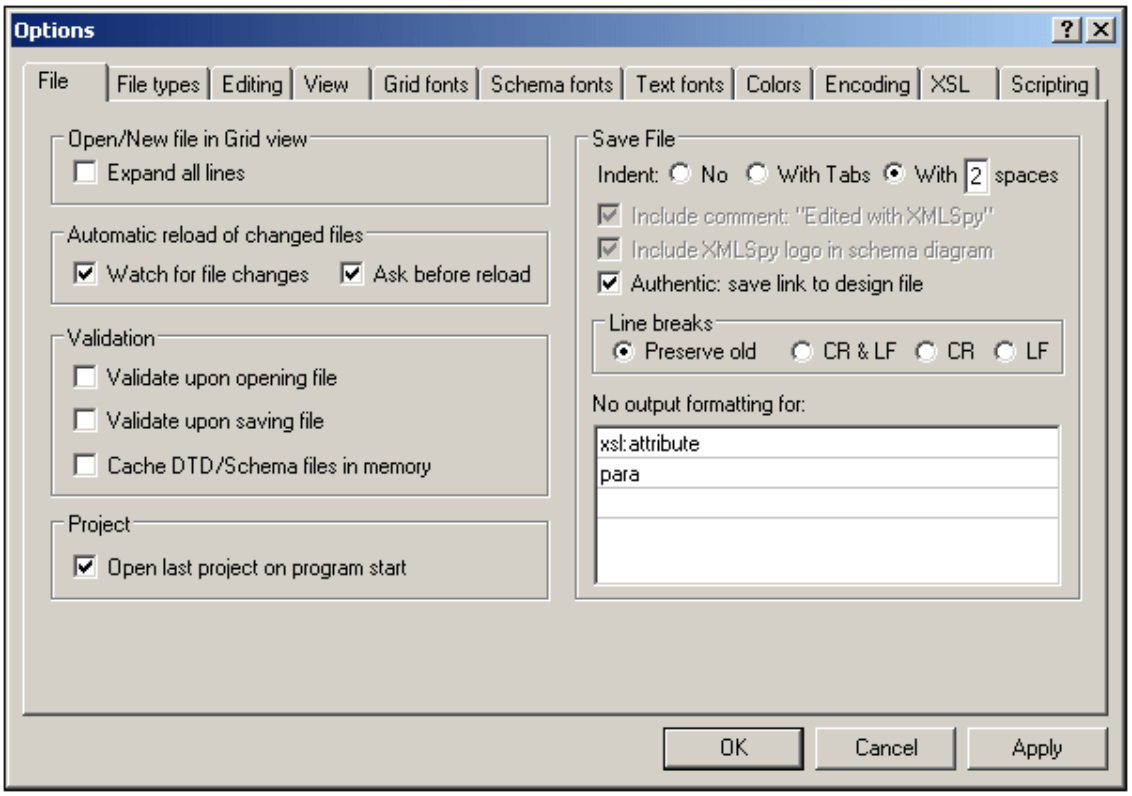
Mit dem Befehl "Optionen" können Sie globale Programmeinstellungen in einem mit Registern versehenen Dialogfeld vornehmen. Diese Einstellungen werden in der Registry gespeichert und auf alle aktuellen und zukünftigen Dokumentfenster angewendet. Die Schaltfläche "Übernehmen" zeigt die Änderungen in den aktuell geöffneten Dokumenten an und speichert die Einstellungen. Die Änderungen sehen Sie sofort in den Fenstern im Hintergrund.

In diesem Abschnitt werden die einzelnen Register des Dialogfelds "Optionen" beschrieben.

Datei

Das Register "**Datei**" definiert die Art, wie XMLSpy 2005 Ihre XML-Dokumente öffnet und speichert. Weitere Einstellungen hierzu finden Sie auf dem Register Kodierung.

© 2005 Altova GmbH



Öffnen/Neue Datei in Grid-Ansicht
Sie können festlegen, ob eine existierende oder eine neue Datei entweder in der Enhanced Grid- oder der Text-Ansicht geöffnet werden soll. Zusätzlich können Sie beim Auswählen der Grid-Ansicht auswählen, ob alle Zeilen ausgeklappt werden sollen.

Auto-laden von geänderten Dateien
Wenn Sie in einer Multi-User Umgebung, oder mit Dateien arbeiten, die von Servern dynamisch erstellt werden, können Sie die Änderungen während des Bearbeitens direkt verfolgen. XMLSpy 2005 wird Sie fragen, ob die Datei neu geladen werden soll, sobald eine Änderung in einem offenen Dokument entdeckt wurde.

Validierung
Wenn Sie Dokument Type Definitions (DTDs) oder Schemas zur Definition der Struktur Ihrer XML-Dokumente verwenden, können Sie das Dokument beim Öffnen oder Speichern automatisch auf Gültigkeit überprüfen lassen.
Mit XMLSpy 2005 können Sie die Dateien auch in den Speicher cachen, sodass unnötiges Neuladen entfällt (z.B. wenn das referenzierte Schema über eine URL aufgerufen wird). Falls in Ihrer schemaLocation-Deklaration eine URL verwendet wird, deaktivieren Sie die Option "DTD/Schema Dateien im Cachespeicher", um am Schema vorgenommene Änderungen sofort anzuzeigen und nicht die im Cache gespeicherte Version des Schemas zu verwenden.

Projekt
Beim Starten von XMLSpy 2005 können Sie das zuletzt verwendete Projekt automatisch anzeigen lassen.

Datei speichern
Beim Speichern einer Datei werden die Einzüge so wie in diesem Fenster angegeben gespeichert, d.h. mit TAB-Zeichen (#x09), zwei Leerzeichen oder ohne Einzug. Wenn ein

StyleVision Power Stylesheet mit einer XML-Datei verknüpft ist, bewirkt die Option "Authentic: Link zu Design-Datei speichern", dass die Verknüpfung zum StyleVision Power Stylesheet mit der XML-Datei gespeichert wird. Beachten Sie, dass der hier defnierte Einzug auch verwendet wird, wenn Sie das Dokument mit Hilfe der Funktion "XML-Text Druckaufbereitung" formatieren (**Bearbeiten | XML-Text Druckaufbereitung**).

Beim Speichern eines XML-Dokuments fügt XMLSpy 2005 im oberen Bereich des Dokuments den kurzen Kommentar `<!-- Mit XMLSpy 2005 bearbeitet http://www.altova.com -->` ein. Dieser Kommentar kann nur von lizenzierten Benutzern deaktiviert werden.

Wenn Sie ein Content Model-Diagramm speichern (über die Menüoption **Schema-Design | Schema-Dokumentation generieren**), inkludiert XMLSpy 2005 das XMLSpy 2005 Logo. Diese Option kann nur von lizenzierten Benutzern deaktiviert werden.

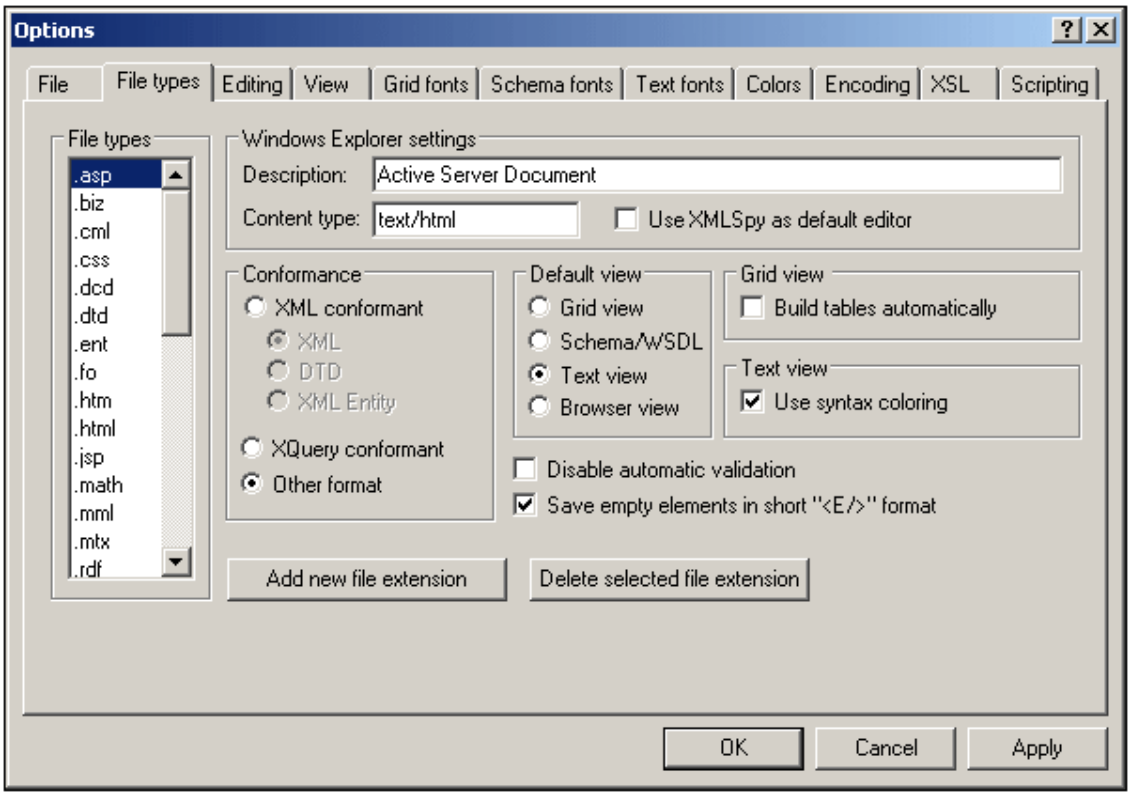
Zeilenumbrüche
Wenn Sie eine Datei öffnen, werden die Zeichen für Zeilenumbrüche beibehalten, wenn Sie die Option **Alte beibehalten** gewählt haben. Alternativ dazu können Zeilenumbrüche in **CR&LF** für PCs, **CR** für MacOS, oder **LF** für Unix kodiert werden.

Keine Ausgabe-Formatierung für
In der Textansicht können Sie Child-Elemente aus Gründen der besseren Übersichtlichkeit einrücken (mit Tabulator oder Leerzeichen). Wenn Sie einen Einzug definiert haben, wird die Datei mit den jeweiligen Whitespaces gespeichert. Der Einzug eines Elements dient auch dazu, seine Position in der Dokumentenhierarchie deutlich zu machen (**Bearbeiten | XML-Text Druckaufbereitung**). Sie können jedoch den (in diesem Dialogfeld oder manuell in der Datei) definierten Einzug für einzelne Elemente auch außer Kraft setzen, indem Sie den Namen des Elements in das Feld **Keine Ausgabe-Formatierung für** eingeben. Alle in diesem Feld definierten Elemente werden ohne Whitespaces zwischen aufeinanderfolgenden untergeordneten Elementen gespeichert (und angezeigt).

Dateiarten

Auf dem Register "Dateiarten" können Sie für XMLSpy 2005 verschiedene Einstellungen für die einzelnen Dateiarten festlegen.

© 2005 Altova GmbH



Wählen Sie eine Dateiart im Listenfeld "Dateiarten" aus, um folgende Funktionen für die ausgewählte Dateiart anzupassen:

Windows Explorer Einstellungen
Sie können die Beschreibug der Dateiart und den MIME-konformen Inhaltstyp, der von Windows Explorer verwendet wird, ändern. Weiters können Sie definieren, ob XMLSpy 2005 als der standardmäßige Editor für diese Dateiart verwendet werden soll.

XML Konformität

XMLSpy 2005 bietet für verschiedene Dateiarten spezifische Editierfunktionen und andere Funktionen, die Sie durch Einstellung der Konformität in dieser Option festlegen können. In XMLSpy 2005 können Sie für eine Dateiart festlegen, dass sie mit XML-, XQuery und anderer (Text)-Syntax konform sein muss. Des Weiteren lässt sich für XML-Dateiarten die Konformität mit XML, DTD and XML Entitäten festlegen. Für viele Dateiarten ist bereits eine standardmäßige Konformitätseinstellung festgelegt, die am besten beibehalten werden sollte, es sei denn Sie fügen eine neue Dateiart hinzu oder möchten für eine bestimmte Dateiart die Konformität explizit ändern.

Standard Ansicht
In dieser Gruppe definieren Sie, welche Standardansicht für welche Dateiart verwendet werden soll.

Grid-Ansicht
Dieses Kontrollkästchen definiert, ob in der Enhanced Grid Ansicht automatisch Tabellen erzeugt werden sollen.

Textansicht
In diesem Textfeld können Sie die Syntaxfärbung für bestimmte Dateiarten bestimmen.

Automatische Validierung deaktivieren
Mit dieser Option können Sie für bestimmte Dateiarten die automatische Validierung deaktivieren. Die automatische Validierung wird normalerweise beim Öffnen oder Speichern einer Datei oder beim Wechseln der Ansicht durchgeführt.

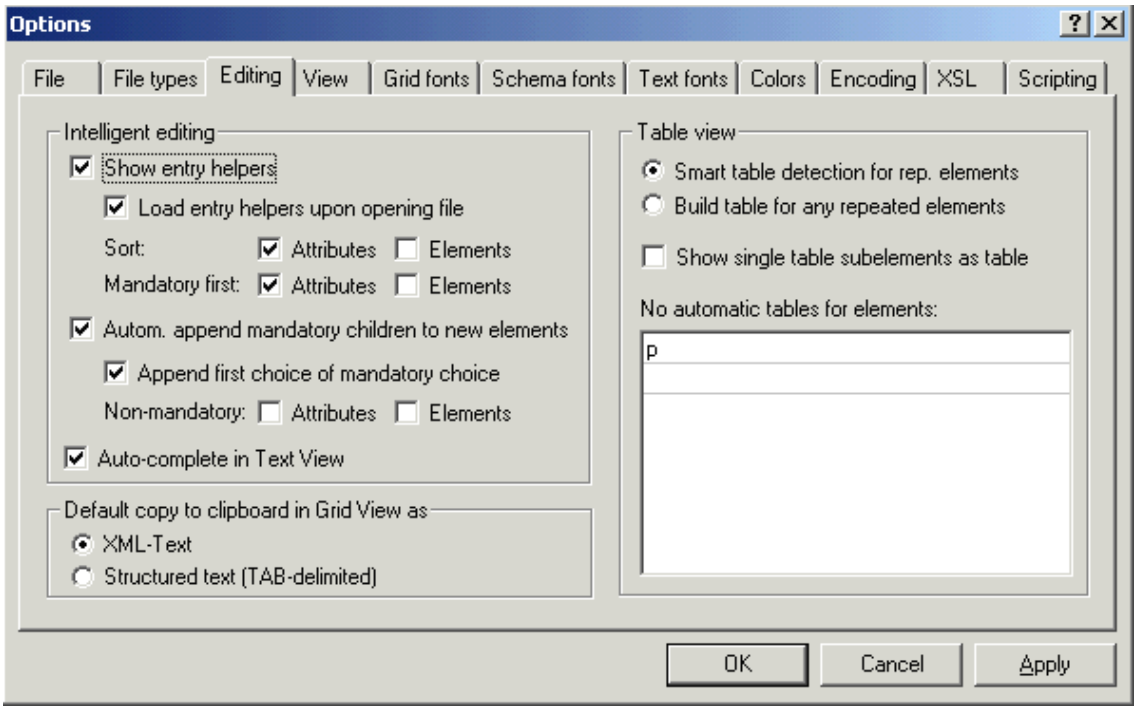
Leere Elemente im Kurzformat "<E/>" speichern
Manche XML-Tools haben Probleme, die Kurzform <Element/> für leere Elemente zu verstehen (die in der XML 1.0 Spezifikation definiert wurde). Sie können XMLSpy 2005 anweisen, diese Elemente in der Langform zu speichern `<Element></Element>`, die gleichermaßen gültig ist.

Schaltfläche "Dateierweiterung hinzufügen"
Eine neue Dateierweiterung wird in die Liste der Dateiarten aufgenommen. Mit den Optionen in diesem Dialogfeld definieren Sie die Einstellungen für die neue Dateiart.

Schaltfläche "Markierte Dateierweiterung löschen"
Löscht die Dateiart und die dazugehörigen Einstellungen.

Bearbeiten

Auf dem Register "**Bearbeiten**" können Sie Bearbeitungseinstellungen für XMLSpy 2005 definieren.



Editier-Hilfen
XMLSpy 2005 bietet Ihnen intelligente Eingabehilfen zum Bearbeiten von Dokumenten. Verschiedene Bearbeitungseinstellungen für diese Eingabehilfen lassen sich hier anpassen.

In die Zwischenablage der Grid-Ansicht kopieren als
Wählen Sie das Format, in dem Daten in Fremdanwendungen exportiert werden sollen. Wenn Sie XML-Text wählen, wird der Inhalt der Zwischenablage genau wie die resultierende XML-Datei formatiert und mit Tags versehen.

Im strukturierten Textmodus wird versucht, den Inhalt der Zwischenablage als Tabelle

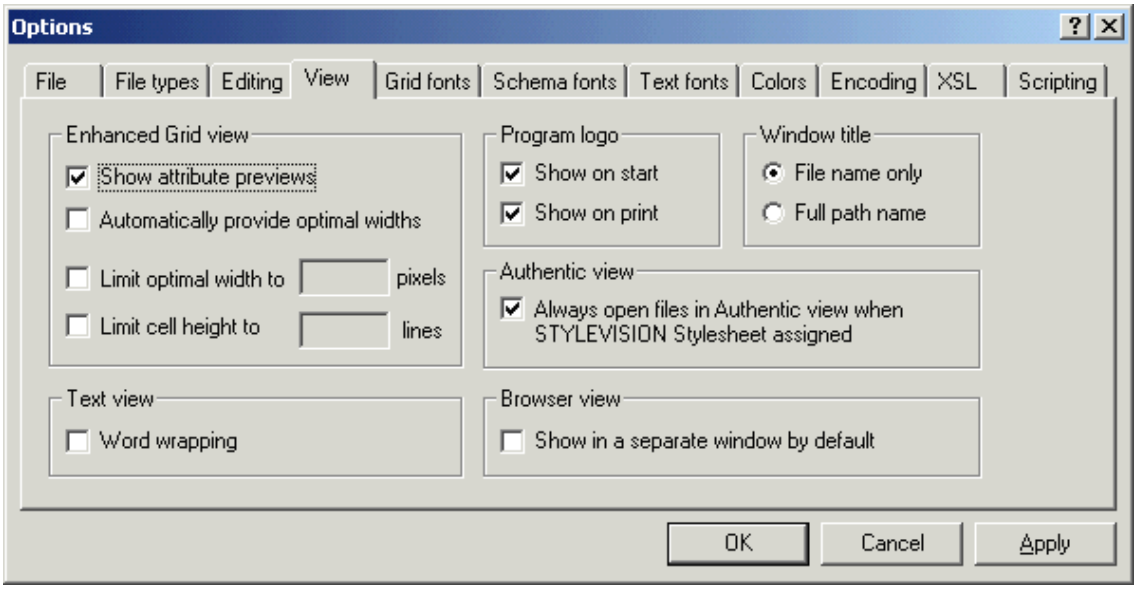
 © 2005 Altova GmbH

(Verwendung in Spreadsheet- oder Datenbankapplikationen) zu formatieren. Diese Option betrifft nicht das interne Zwischenablagenformat, das XMLSpy 2005 zum Kopieren und Einfügen verwendet.

Tabellen-Ansicht
Weiters können Sie einstellen, wie XMLSpy 2005 sich wiederholende Elemente in der Tabellenansicht darstellen soll.

Ansicht

Auf dem Register "Ansicht" können Sie die Darstellung der XML-Dokumente in XMLSpy 2005 anpassen.



Enhanced Grid-Ansicht
Reduzierte Elemente in der Enhanced Grid Ansicht können als Vorschau angezeigt werden. Attribute und deren Werte werden in Grau in der gleichen Zeile wie das zugehörige Element dargestellt. Sie können beim Bearbeiten der Dateien automatisch die optimale Breite anwenden und die maximale Höhe und Breite der Spalte festlegen.

Text-Ansicht
Deaktivieren Sie den Zeilenumbruch in der Text-Ansicht.

Programmlogo
Das Programmlogo beim Starten kann deaktiviert werden, um die Applikation schneller starten zu können.

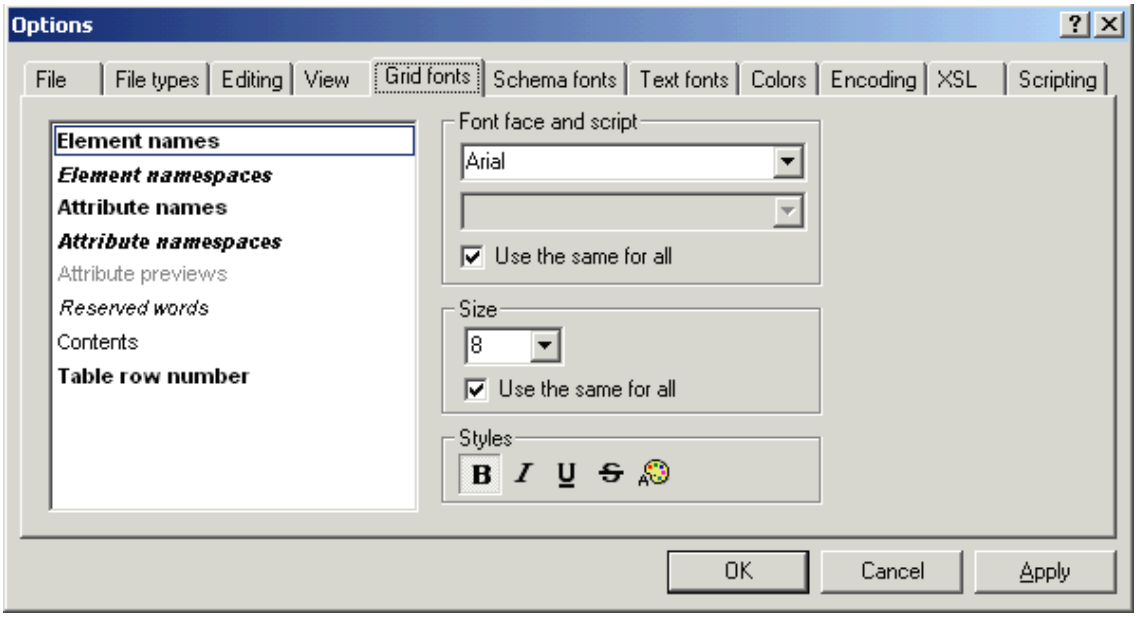
Titelleiste-Inhalt
Im Fenstertitel jedes Dokumentfensters kann entweder nur der Dateiname oder der komplette Pfad angezeigt werden.

Authentic-Ansicht
XML-Dateien basierend auf einem **StyleVision Power Stylesheet** (*.sps), werden bei Aktivieren dieser Option automatisch in der Authentic-Ansicht geöffnet.

Browser-Ansicht
Sie können die Browser-Ansicht in einem separaten Fenster öffnen, sodass Sie die Ansicht, in der Sie arbeiten, und die Browser-Ansicht nebeneinander sehen.

Grid-Schriftarten

Auf dem Register **Grid-Schriftarten** können Sie das Aussehen von Texten in der Enhanced Grid-Ansicht anpassen.



Schriftart und Schrift auswählen
Wählen Sie Schriftart und Schriftgröße, die in der Enhanced Grid Ansicht verwendet werden sollen. Dieselben Einstellungen werden auch für das Drucken verwendet. Deswegen können nur **TrueType** Schriftarten ausgewählt werden.

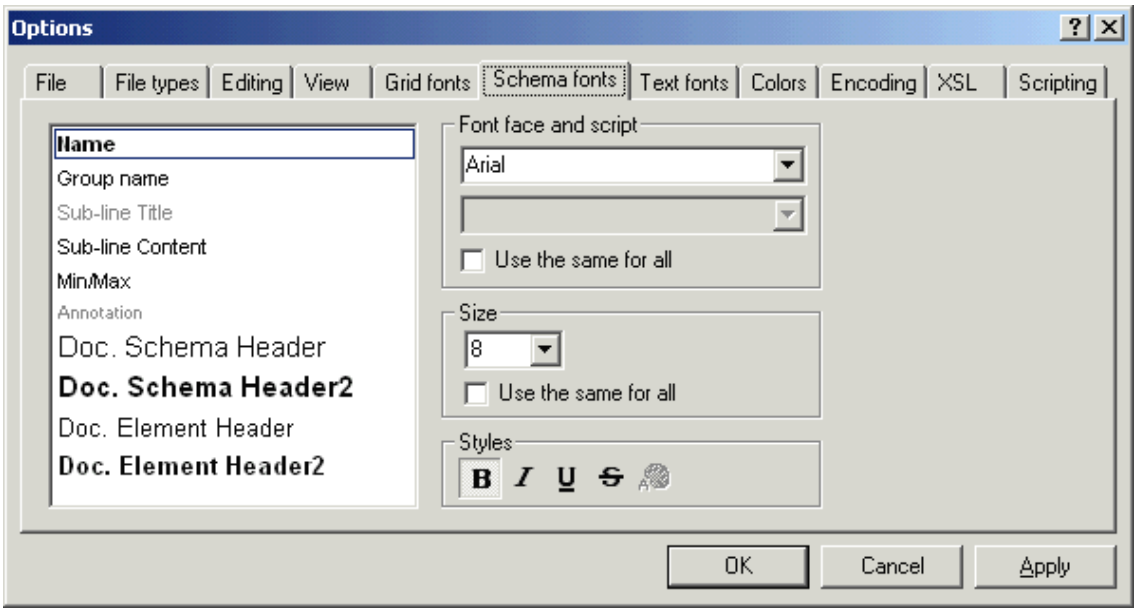
Größe
Wählen Sie hier die gewünschte Größe aus. Wenn Sie für alle Elemente die selbe Schriftgröße verwenden möchten, aktivieren Sie das Kontrollkästchen "Auf alle anwenden".

Stil
Stil und Farbe können im über die Auswahlschaltflächen eingestellt werden. Die Einstellungen werden sofort als Vorschau in der in Liste auf der linken Seite angezeigt. So sehen Sie sofort, wie Ihr Dokument aussehen wird.

Schemaschriftarten

Auf dem Register **Schemaschriftarten** können Sie das Aussehen von Texten in der Schema/WSDL-Ansicht anpassen.

© 2005 Altova GmbH



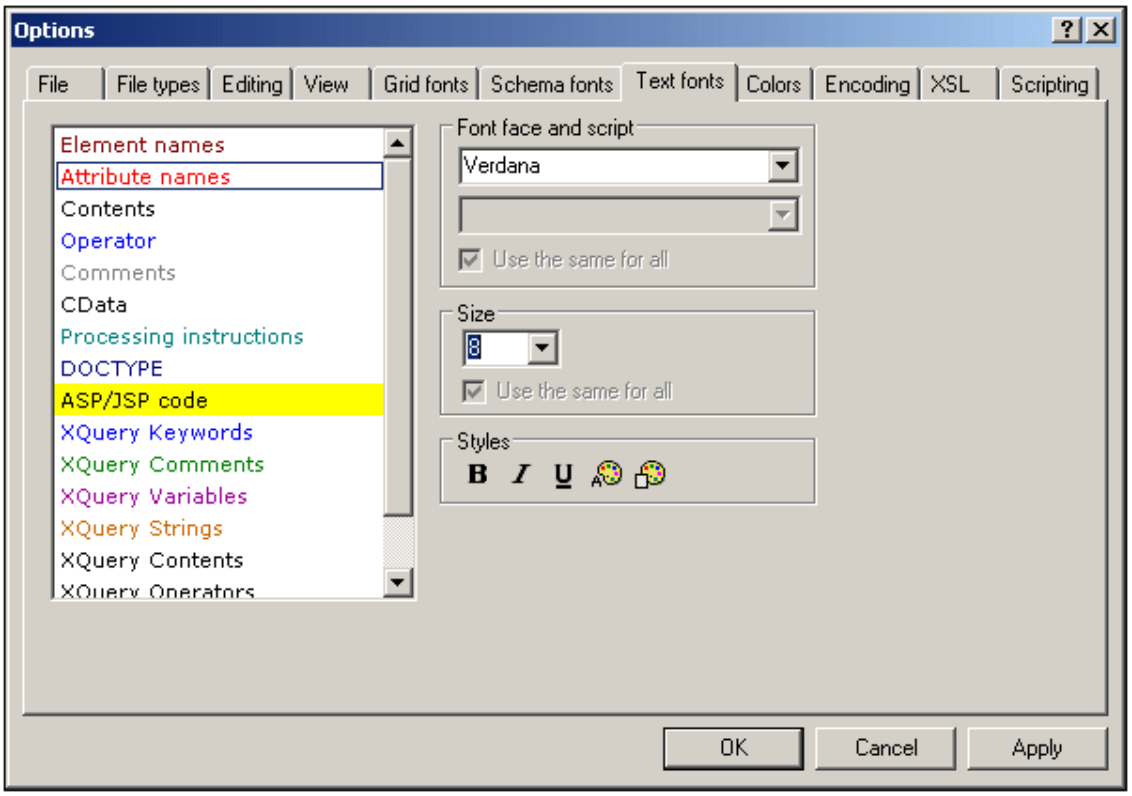
Schriftart und Schrift auswählen
Wählen Sie Schriftart und Schriftgröße, die in der Schema/WSDL Design-Ansicht verwendet werden sollen. Dieselbe Schriftgröße und Schriftart wird zum Drucken und Erstellen von Schema Dokumentation verwendet. Deswegen können nur TrueType Schriftarten ausgewählt werden. Komponenten mit dem Präfix "Doc." werden in der Schema-Dokumentation verwendet.

Größe
Wählen Sie hier die gewünschte Größe aus. Wenn Sie für alle Elemente die selbe Schriftgröße verwenden möchten, aktivieren Sie das Kontrollkästchen "Auf alle anwenden".

Stil
Stil und Farbe können über die Auswahlschaltflächen eingestellt werden. Die Einstellungen werden sofort als Vorschau in der in Liste auf der linken Seite angezeigt. So sehen Sie sofort, wie Ihr Dokument aussehen wird.

Textschriftarten

Auf dem Register **Textschriftarten** können Sie das Aussehen von Texten in der Text-Ansicht anpassen.

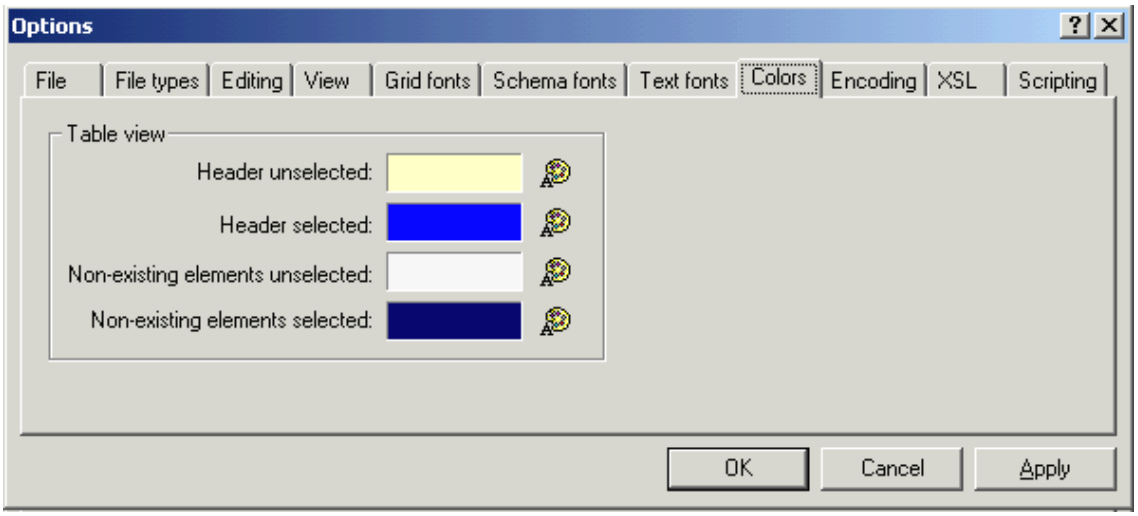


Die im linken Bereich aufgelisteten Typen sind XML-Node-Tpyen, ASP/JSP-Code und XML-Syntax-Komponenten. Sie können die allgemeine Schriftart, den Stil und die Größe des in der Textansicht gezeigten Texts definieren. Beachten Sie, dass für alle Textarten dieselbe Schriftart, derselbe Stil und dieselbe Größe verwendet werden. Sie können für einzelne Textarten nur Textfarbe und Hintergrundfarbe ändern. Dies aktiviert die Syntaxfärbung.

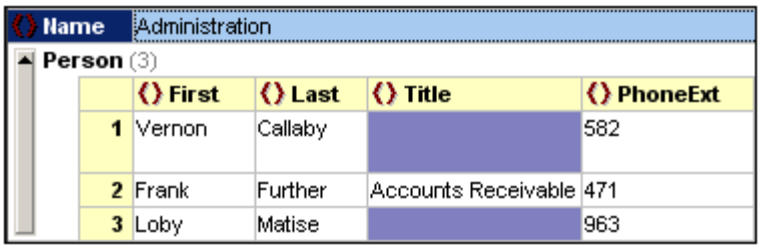
Farben

Auf dem Register **Farben** können Sie die in der Tabellen- oder Grid-Ansicht verwendeten Farben anpassen.

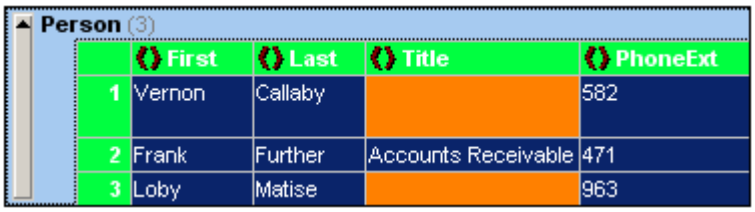
© 2005 Altova GmbH



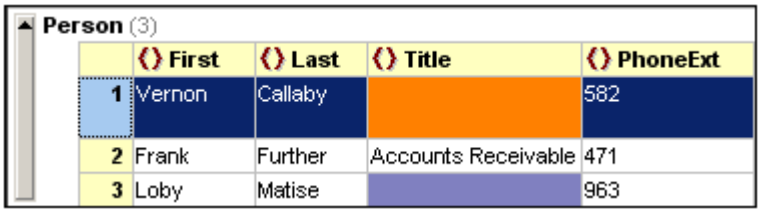
Farben der Tabellenüberschrift
Die Optionen "Tabellenbezeichnung nicht aktiv" und "Tabellenbezeichnung aktiv" beziehen sich auf die Spalten- und Zeilenüberschriften. Im unten gezeigten Screenshot sind die Tabellenbezeichnungen deaktiviert. Die Farbe entspricht den Einstellungen im obigen Dialogfeld.



Die im Feld "Tabellenbezeichnungen aktiv" definierte Farbe wird aktiviert, wenn alle Überschriften ausgewählt sind (*Screenshot unten*) - nicht aber, wenn nur einzelne Überschriften ausgewählt werden. Sie sehen dies im nachfolgenden Screenshot, in dem die im obigen Dialogfeld definierten Farben verwendet werden. Die Auswahl aller Tabellenüberschriften erfolgt durch Klicken auf die Zelle, in der sich beide Überschriftszeilen schneiden oder durch Auswahl des Elements, das als Tabelle dargestellt wird, bzw. durch Auswahl eines seiner übergeordneten Elemente.



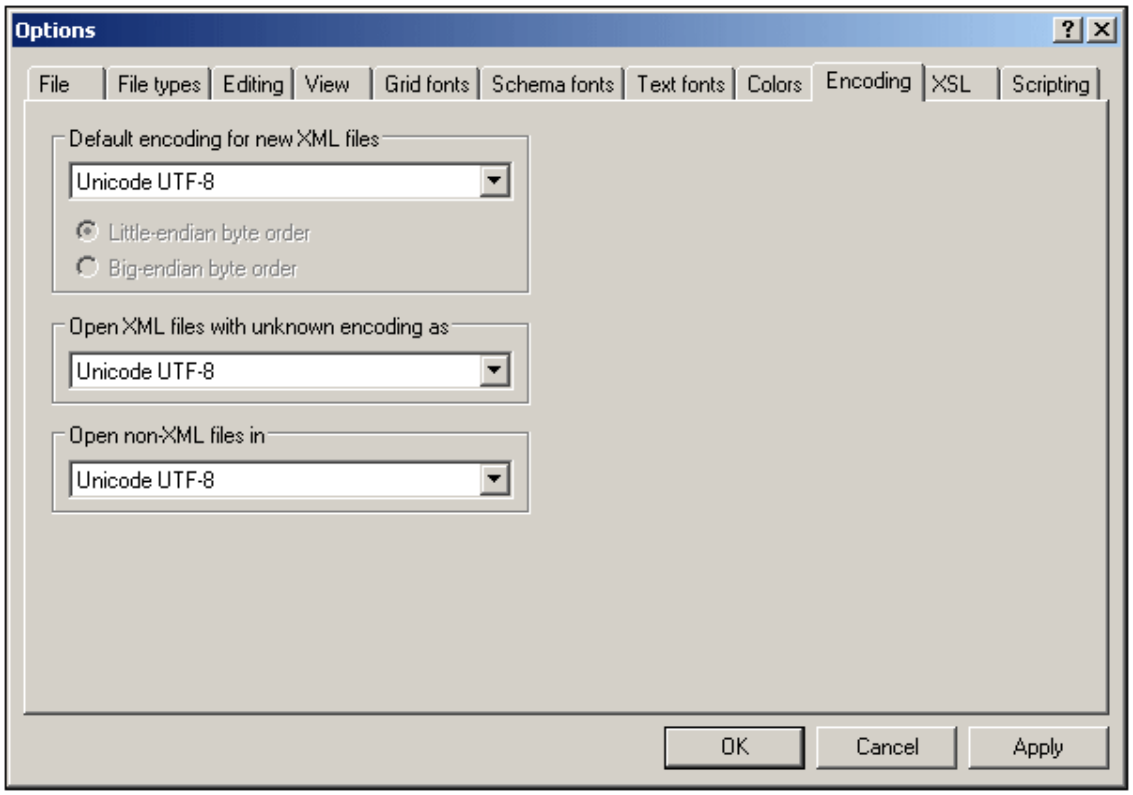
Nicht-existente Elemente
Wenn eine Zelle in einer Tabelle leer ist, existiert das durch diese Zelle dargestellte Element bzw. Attribut im XML-Dokument nicht. Man kann solchen nicht-existenten Zellen eine andere Hintergrundfarbe zuweisen, wenn sie ausgewählt oder nicht ausgewählt sind. Ein Beispiel dafür sehen Sie im nachfolgenden Screenshot, in welchem die erste Zeile ausgewählt ist.



Hinweis: Zusätzlich zu den hier definierten Farben, verwendet XMLSpy 2005 die normalen Einstellungen für Menüfarben und hervorgehobenen Text, die in der Systemsteuerung Ihrer Windowsinstallation unter Anzeigeeinstellungen festgelegt sind.

Kodierung

Auf dem Register **Kodierung** definieren Sie die Einstellungen für die Dateikodierung.



Standardkodierung für neue XML-Dateien
Die Standardkodierung für neue Dateien kann im Dialogfeld "Optionen" definiert werden, sodass jedes neue Dokument automatisch mit der entsprechenden XML-Deklaration und der hier gewählten Kodierungsspezifikation erstellt wird. Wird eine 2- oder 4-Byte-Kodierung als standardmäßige Kodierung ausgewählt (z.B. UTF-16, UCS-2 oder UCS-4) können Sie weiters wählen, ob die Byteordnung für XML-Dokumente Little Endian oder Big Endian sein soll. Die Einstellung "Standardkodierung für neue XML-Dateien" funktioniert nur bei Dateien, denen kein StyleVision Power Stylesheet zugewiesen wurde.

Die Kodierung für existierende Dateien wird immer beibehalten und kann nur mit dem Befehl **Datei | Kodierung** geändert werden.

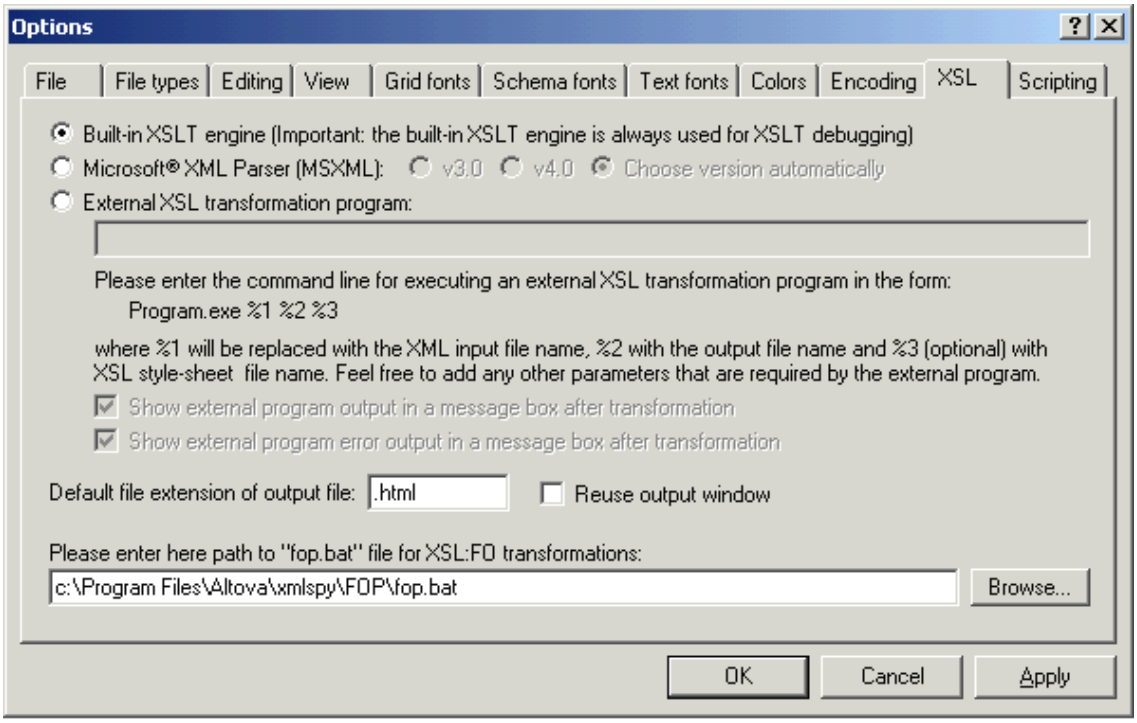
© 2005 Altova GmbH

Dateien mit unbekannter Kodierung öffnen als
Sie können auswählen, mit welcher Kodierung eine Datei geöffnet werden soll, die keine Kodierungsspezifikation enthält oder deren Kodierung nicht automatisch bestimmt werden konnte. Beachten Sie, dass XML-Dateien, für die keine Kodierung festgelegt wurde, mit der korrekten Kodierung UTF-8 gespeichert werden.

Nicht-XML-Dateien öffnen in
Sie können auswählen, welche Kodierung für nicht XML-Dateien verwendet werden soll, die Sie bearbeiten möchten.

XSL

Auf dem Register **XSL** werden die Einstellungen für XSLT-Transformationen sowie der Pfad zu einem FO-Prozessor definiert.



XMLSpy 2005 enthält den Altova XSLT 1.0 Prozessor und den Altova XSLT 2.0 Prozessor, die Sie für XSLT-Transformationen verwenden können. Je nach dem Wert des `version`-Attributs im Element `xsl:stylesheet` oder `xsl:transform` wird immer der jeweilige XSLT-Prozessor für Transformationen und in Verbindung mit dem XSLT/XQuery Debugger verwendet.

Zur Transformation von XML-Dokumenten kann entweder der MSXML 3.0- oder 4.0 Parser (vorinstalliert), sowie jeder beliebige externe XSLT Prozessor verwendet werden.
Beachten Sie, dass Parameter, die im Dialogfeld "XSLT-Eingabeparameter" eingestellt wurden, nur an die internen Altova XSLT-Prozessoren übergeben werden und nicht an andere XSLT-Prozessoren, die Sie als Standardprozessor für die Verwendung mit XMLSpy 2005 definiert haben.

Microsoft XML Parser (MSXML) 3.0 oder 4.0
Wenn Sie Internet Explorer 5 oder höher (z.B. mit dem MSXML Modul) für die Verarbeitung von XSLT verwenden, müssen Sie angeben, ob die resultierende Ausgabe als XML-konformes Dokument (HTML oder WML) oder als nicht-XML konforme Textdatei transformiert werden soll.

Die Option "Version automatisch wählen" ist standardmäßig aktiviert.
XMLSpy 2005 versucht zuerst Version 4.0 zu verwenden und falls diese nicht verfügbar ist, Version 3.0. Sie können die Version jedoch auch manuell durch Aktivieren des entsprechenden Optionsfelds auswählen.

Externes XSL Transformationsprogramm
Wenn Sie einen externen XSL Prozessor verwenden, müssen Sie angeben, welche Befehlszeile als String verwendet werden soll, damit XMLSpy 2005 die XSL-Transformation initiieren kann. In diesem Fall verwenden Sie eine der folgenden Variablen um den String für die Befehlszeile zu erstellen:

%1 XML Instanzdokument, welches verarbeitet werden soll
%2 Ausgabedatei, die erzeugt werden soll
%3 XSL Stylesheet, das verwendet werden soll (optional, wenn das Dokument eine <?xsl-Stylesheet ...?> Referenz enthält)

Der Befehl zum Ausführen einer einfachen Transformation mit dem Saxon (XSLT 1.0) Prozessor lautet z.B.:

```
saxon.exe -o output.xml input.xml stylesheet.xslt parameter-
   name=parameter-value
```

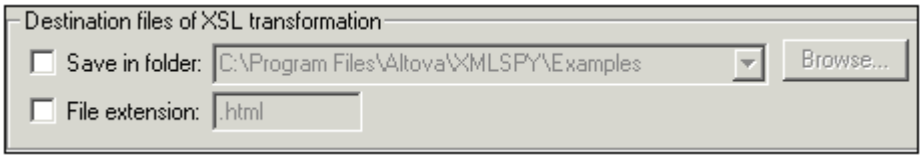
Um XMLSpy 2005 für die Arbeit mit dem Saxon-Prozessor zu konfigurieren, aktivieren Sie das Optionsfeld "Externes XSL-Transformationsprogramm" und geben Sie im Textfeld Folgendes ein:

```
c:\saxon\saxon.exe -o %2 %1 %3 parameter-name=parameter-value
```

Weiters können Sie auswählen, ob das Resultat und Fehlermeldungen des externen XSLT Prozessors in XMLSpy 2005 in einem Nachrichtenfeld angezeigt werden soll.

Ausgabefenster wiederverwenden:
Ist diese Option deaktiviert, wird sie nur dann angewendet, wenn unter den Ordnereigenschaften auch der Ausgabedateipfad der zugehörigen XML-Datei unter "Speichern in" in einem XMLSpy 2005-Projekt gleichermaßen deaktiviert wurde.

Fenster Ordner-Eigenschaften:

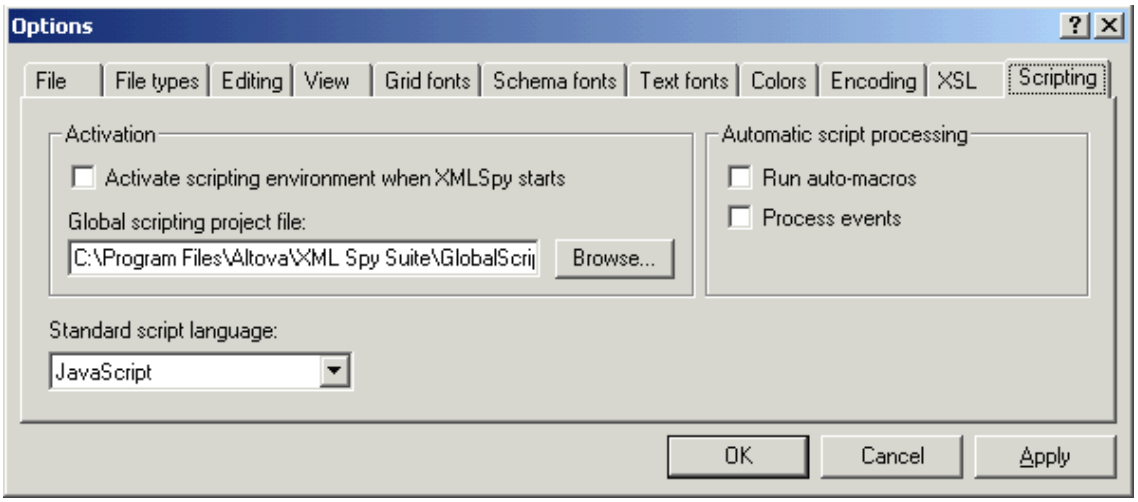


FO-Prozessor
Wenn Sie den Befehl für die FO-Transformation verwenden wollen, müssen Sie den Pfad zur ausführbaren Datei des FO-Prozessors in diesem Feld festlegen.

Skripting-Umgebung

Auf dem Register **Skripting-Umgebung** können Sie die Skripting-Umgebung, sowie weitere spezifische Skriptparameter definieren.

© 2005 Altova GmbH



Skripting-Umgebung:
Definieren Sie, ob die Skriptumgebung beim Starten von XMLSpy 2005 aktiviert werden soll. Klicken Sie auf die Schaltfläche "Durchsuchen...", um die Skripting-Projektdatei auszuwählen.

Standard Skriptsprache:
Wählen Sie die Skriptsprache, die verwendet werden soll: JavaScript oder VBScript.

Automatische Skriptverarbeitung:
In diesem Abschnitt definieren Sie, ob automatische Makros laufen dürfen, und ob die Abarbeitung von Events erlaubt sein soll.

13.15 Menü "Fenster"

Mit Hilfe dieses Menüs können Sie die Dokumentfenster für Ihre XMLSpy 2005 Sitzung anordnen. Das Menü "Fenster" enthält Standardbefehle, die aus den meisten Windows-Applikationen bekannt sind.

Sie können die offenen Dokumente hintereinander oder nebeneinander anzeigen. Weiters können Sie Dokumentsymbole neu anordnen (nachdem Sie diese minimiert haben) und die verschiedenen Eingabehilfenfenster ein- oder ausblenden oder direkt über das Menü in ein beliebiges offenes Dokument-Fenster wechseln.

13.15.1 Überlappend

Mit diesem Befehl arrangieren Sie alle offenen Dokumentfenster, sodass sie übereinander und überlappend angezeigt werden.

13.15.2 Horizontal anordnen

Mit diesem Befehl arrangieren Sie alle offenen Dokumente **horizontal nebeneinander**, sodass alle Dokumentfenster gleichzeitig sichtbar sind.

13.15.3 Vertikal anordnen

Mit diesem Befehl arrangieren Sie alle offenen Dokumente **vertikal nebeneinander**, sodass alle Dokumentfenster gleichzeitig sichtbar sind.

13.15.4 Projekt-Fenster

Mit diesem Befehl schalten Sie das Projekt-Fenster ein oder aus.

Dieses Fenster ist abdockbar, d.h. wenn Sie auf die Titelleiste klicken, können Sie das Fenster an jede beliebige Stelle verschieben. Wenn Sie mit der rechten Maustaste auf die Titelleiste klicken, können Sie das Fenster andocken oder ausblenden.

13.15.5 Info-Fenster

Mit diesem Befehl schalten Sie das Info-Fenster ein oder aus.

Dieses Fenster ist andockbar, d.h. wenn Sie auf die Titelleiste klicken, können Sie das Fenster an jede beliebige Stelle verschieben. Mit Rechtsklick docken Sie das Fenster wieder an, oder verbergen es.

13.15.6 Eingabehilfen

Mit diesem Befehl blenden Sie alle drei Eingabehilfe-Fenster ein oder aus.

Dieses Fenster ist andockbar, d.h. wenn Sie auf die Titelleiste klicken können Sie das Fenster an jede beliebige Stelle verschieben. Mit Rechtsklick docken Sie das Fenster wieder an, oder verbergen es.

© 2005 Altova GmbH

13.15.7 Alle ein/aus

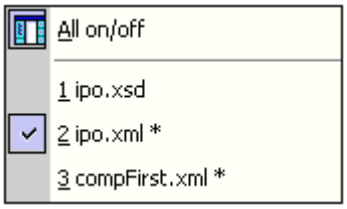
Mit diesem Befehl werden alle andockbaren Fenster zusammen ein- oder ausgeschaltet:

- Projekt-Fenster
- Info-Fenster
- alle drei Eingabehilfen-Fenster

Diese Funktion ist besonders hilfreich, wenn Sie schnell alle anderen Fenster ausblenden möchten, um Ihr Dokument in maximaler Größe anzeigen zu können.

13.15.8 Liste aller offenen Fenster

In dieser Liste werden alle derzeit offenen Fenster angezeigt. So können Sie schnell von einem Dokument zum anderen wechseln.



Hierzu können Sie auch die Tastaturkürzel Strg-TAB oder Strg F6 verwenden.

13.16 Menü "Hilfe"

Im Hilfemenü finden Sie alle Befehle, die Sie benötigen, um weitere Informationen zu spezifischen Themen von XMLSpy 2005 zu erhalten, sowie Links zu Informationen und Supportseiten auf unserem Webserver.

Im Hilfemenü finden Sie weiters den Registrierungsdialog, über den Sie Ihren Keycode eingeben können, sobald Sie das Produkt erworben haben.

13.16.1 Inhaltsverzeichnis...

Mit diesem Befehl erhalten Sie eine **hierarchische Darstellung** aller Kapitel und Themen, die in der Onlinehilfe enthalten sind. Verwenden Sie diesen Befehl, um direkt von XMLSpy 2005 in das Inhaltsverzeichnis zu gelangen.

Sobald das Hilfefenster geöffnet wurde, können Sie zwischen dem Inhaltsverzeichnis, dem Index, und der Suchfunktion wählen. Auf dem Register "Favoriten" können Sie Lesezeichen zu Kapiteln innerhalb der Hilfe setzen.

13.16.2 Index...

Mit diesem Befehl gelangen Sie in das **Stichwortverzeichnis** der Onlinehilfe. Sie können auch das Register Index in der Onlinehilfe verwenden.

Der Index bietet eine Liste von Stichwörtern, mit denen Sie durch Doppelklick zu den jeweiligen Abschnitten gelangen, die das Stichwort behandeln. Wenn es mehrere Abschnitte oder Kapitel gibt, in denen das Stichwort vorkommt, werden Sie aufgefordert, aus einer Liste das gewünschte Kapitel auszuwählen.

13.16.3 Suche...

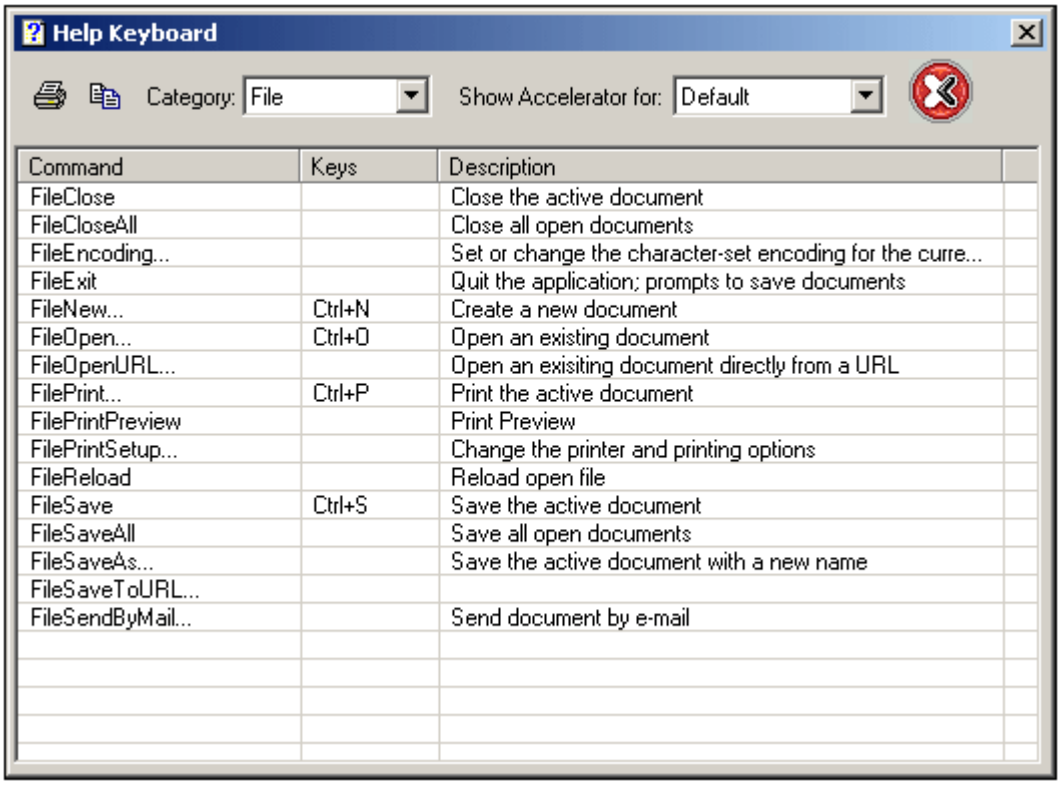
Mit dem Suchbefehl können Sie eine **Volltextsuche** in der gesamten Onlinehilfe starten.

- Geben Sie den Suchbegriff im Suchfeld ein und drücken Sie die Eingabetaste. Die Onlinehilfe zeigt eine Liste von Kapiteln an, in denen der Suchbegriff vorkommt.
- Doppelklicken Sie auf einen Eintrag in der Liste, um das entsprechende Kapitel aufzurufen.

13.16.4 Tastaturzuordnungen...

Bei Auswahl des Befehls **Tastenzuordnungen...** wird eine Infobox angezeigt, in der alle Befehle in XMLSpy 2005 nach Menüs geordnet angezeigt werden. Die Menübefehle werden mit einer Beschreibung und dem entsprechenden Tastaturkürzel für den jeweiligen Befehl angezeigt.

© 2005 Altova GmbH



Um die Befehle eines bestimmten Menüs anzuzeigen, wählen Sie den Menünamen in der Dropdown-Liste "Kategorie" aus. Durch Klicken auf das Druckersymbol können Sie den Befehl ausdrucken.

13.16.5 Registrieren...

Wenn Sie XMLSpy 2005 zum ersten Mal starten, wird automatisch das Registrierungsdialogfeld angezeigt. Hier können Sie Ihr Softwareprodukt registrieren, sodass Sie die Supportdienste nutzen können, die lizenzierten Benutzern zur Verfügung stehen. Um die Software verwenden zu können, geben Sie den gelieferten Keycode in das Dialogfeld ein.

Kostenlose Evaluierungsversion
Wenn Sie XMLSpy 2005 von unserem Webserver heruntergeladen haben, und Sie die **KOSTENLOSE** 30-Tage Evaluierungsversion aktivieren möchten, klicken Sie auf "KOSTENLOSEN Evaluierungs-Key" und geben Sie uns Ihren Namen, Ihre Firma und Email-Adresse bekannt. XMLSpy 2005 sendet diese Informationen über Ihre Internet-Verbindung an unseren Webserver. Hier wird ein Evaluierungslizenz-Keycode generiert, mit dem Sie die Software aktivieren können, und der an die **von Ihnen angegebene Email-Adresse** gesendet wird.

Nachdem Sie die Lizenz angefordert haben, kontrollieren Sie bitte den Posteingang Ihrer Emails. Die Lizenz-Email sollte innerhalb von kurzer Zeit bei Ihnen einlangen.

Sollten Sie keine Lizen erhalten, kann es sein, dass der Vorgang aufgrund von Firewallrestriktionen innerhalb Ihres Netzwerkes nicht durchgeführt werden konnte. In diesem Fall senden Sie bitte eine E-Mail an unser Support-Team, welches Ihnen einen Keycode zusenden wird.

Nachdem Sie Ihren Keycode erhalten haben, geben Sie diesen im Keycode-Feld des Registrierungs-Dialogfeldes ein, und klicken Sie auf OK um XMLSpy 2005 zu starten.

Wann immer Sie eine lizenzierte Version von XMLSpy 2005 bestellen möchten, erreichen Sie über die Schaltfläche "Permanenten Key-Code ERWERBEN..." oder über das Bestellformular unseren sicheren Online-Shop im Internet.

Lizenzierte Version

Wenn Sie eine *Einzellizenz* für XMLSpy 2005 bestellt haben, erhalten Sie eine E-Mail-Nachricht mit Ihrem Namen, Ihrer Firma und dem Keycode. Bitte geben Sie diese Daten im Registrierungsdialog **genauso** ein, wie sie in der E-Mail angegeben wurden, da ansonsten die Software nicht aktiviert werden kann.

Wenn Ihre Firma eine *Multiuserlizenz* für XMLSpy 2005 erworben hat, erhalten Sie eine E-Mail-Nachricht mit dem Firmennamen, Ihrem Namen und dem Keycode. Bitte geben Sie diese Daten im Registrierungsdialog **genauso** ein, wie sie in der E-Mail angegeben wurden, da ansonsten die Software nicht aktiviert werden kann.

Bitte beachten Sie, dass die XMLSpy 2005 Lizenzvereinbarung es nicht gestattet, mehr als die maximale Anzahl an lizenzierten Kopien von XMLSpy 2005 auf Computern in Ihrer Firma zu installieren.

13.16.6 Bestellformular...

Wenn Sie eine lizenzierte Version von XMLSpy 2005 erwerben möchten, verwenden Sie bitte diesen Befehl oder die Schaltfläche "Permanenten Key-Code ERWERBEN..." im Registrierungsdialog. Sie werden dann zu unserem sicheren Online-Shop weitergeleitet. Hier können Sie zwischen verschiedenen Einzel- oder Multiuser-Lizenzpaketen wählen.

Zur Bezahlung können Sie zwischen Zahlung per Kreditkarte, Scheck per Post oder Banküberweisung wählen.

13.16.7 Support Center...

Falls Sie Fragen zu unseren Produkten haben, kontaktieren Sie über diesen Befehl unser Support-Team. Hier finden Sie Links zu unserer Liste von Fragen und Antworten, Supportformulare und die Email-Adressen, um unser Support-Team zu kontaktieren.

13.16.8 Fragen und Antworten im Web...

Um Ihnen den bestmöglichen Support bieten zu können, haben wir eine Liste von Fragen und Antworten im Internet (FAQ) zusammengestellt. Diese Liste wird ständig von unseren Support-Mitarbeitern aktualisiert.

Sollten Sie Probleme mit einem unserer Produkte haben, raten wir Ihnen, zuerst die Fragen und Antworten im Internet zu konsultieren. Oft erhalten Sie hierdurch schnelle Hilfe.

Leider bieten wir derzeit noch keinen telefonischen Support. Ihre E-Mails an das Support-Team werden aber in der Regel innerhalb eines Werktages beantwortet.

Wenn Sie uns Anregungen oder Vorschläge zu XMLSpy 2005 unterbreiten möchten, verwenden Sie bitte unseren Fragebogen.

13.16.9 Komponenten downloaden...

Hier können Sie derzeit den neuesten Microsoft XML Parser, sowie alternative XSLT Transformationssysteme downloaden. Dieser Bereich wird in Zukunft erweitert werden.

© 2005 Altova GmbH

13.16.10 XMLSpy im Internet...

Mit diesem Befehl gelangen Sie zum Altova Webserver auf http://www.altova.com. Hier erhalten Sie alle Informationen, Produkt-Updates und weitere Angebote von Altova.

13.16.11 XMLSpy Training...

Mit diesem Befehl gelangen Sie zum Altova Webserver auf http://www.altova.com. Hier finden Sie Informationen über unsere autorisierten Schulungspartner, die Kurse für XMLSpy 2005 und Advanced XML Application Development (AXAD) anbieten.

13.16.12 Über XMLSpy...

Mit diesem Befehl wird der XMLSpy 2005 Splashscreen und das Dialogfeld mit urheberrechtlichen Informationen angezeigt.

In diesem Dialogfeld wird die Versionsnummer angezeigt. Der Build sowie die komplette Versionsnummer der Version, die Sie verwenden, werden in der Statusleiste angezeigt.

13.17 Altova XSLT 1.0-Prozessor

Der Altova XSLT 1.0-Prozessor entspricht den Vorgaben der XSLT 1.0 Recommendation vom 16. November 1999 mit den folgenden Einschränkungen.

Whitespaces im XML-Dokument
Standardmäßig entfernt der Altova XSLT 1.0 Prozessor alle Whitespaces in Nodes, die nur Whitespaces enthalten, aus dem XML-Quelldokument. Beachten Sie, dass das Vorhandensein, bzw. Fehlen von Nodes, die nur Whitespaces enthalten, einen Einfluss auf den Wert hat, den die `position()`-Funktion zurückgibt.

Unterstützung von XSLT 1.0-Elementen
Nachfolgend sehen Sie eine Liste von XSLT 1.0-Elementen, die nur beschränkt unterstützt werden.

Elemente	Eingeschränkte Unterstützung
`xsl:strip-space`	Nicht unterstützt.
`xsl:preserve-space`	Nicht unterstützt.

© 2005 Altova GmbH

13.18 Altova XSLT 2.0-Prozessor

Der Altova XSLT 2.0-Prozessor entspricht den Vorgaben des XSLT 2.0 Working Draft vom 11. Februar 2005 mit den folgenden Einschränkungen.

Namespaces
Damit Sie die in XSLT 2.0 verfügbaren Typkonstruktoren und Funktionen verwenden können, sollten in Ihrem XSLT 2.0 Stylesheet die folgenden Namespaces deklariert werden. Konventionsgemäß werden die unten angegebenen Präfixe verwendet. Sie können auf Wunsch andere Präfixe verwenden.

Namespace-Name	Präfix	Namespace URI
XML-Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0-Funktionen	`fn:`	`http://www.w3.org/2005/02/xpath-functions`
XPath-Datentypen	`xdt:`	`http://www.w3.org/2005/02/xpath-datatypes`

Normalerweise werden diese Namespaces im Element `xsl:stylesheet` oder `xsl:transform` deklariert, wie in der nachfolgenden Liste gezeigt:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2004/10/xpath-functions"
   xmlns:xdt="http://www.w3.org/2004/10/xpath-datatypes">
   ...
</xsl:stylesheet>
```

Beachten Sie bitte die folgenden Punkte:

- Der Altova XSLT 2.0-Prozessor verwendet den Namespace für die XPath 2.0-Funktionen als Standard-Funktions-Namespace. Sie können daher XPath 2.0 Funktionen und XSLT 2.0 Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet explizit deklarieren, können Sie zusätzlich das in der Deklaration zugewiesene Präfix verwenden.
- Bei Verwendung von Typkonstruktoren und Typen aus den Namespaces für XML-Schema und XPath-Datentypen, müssen die in den Namespace-Deklarationen verwendeten Präfixe verwendet werden.

Bei Zuweisung eines falschen Namespaces wird eine Fehlermeldung ausgegeben. Beachten Sie allerdings, dass einige XPath 2.0-Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` sind definiert). Wenn Sie daher den XPath-Ausdruck `string('Hello')` verwenden, sollte aus der obigen Beschreibung klar hervorgehen, dass, obwohl Sie beabsichtigten den Typkonstruktor `xs:string` zu verwenden, kein Fehler ausgegeben wird, da der Ausdruck als `fn:string('Hello')` evaluiert wird.

Hinweis zu XSLT 2.0-Funktionen: Der Altova XSLT 2.0-Prozessor löst XSLT 2.0-Funktionen ohne Präfix automatisch zum Namespace für XPath 2.0-Funktionen auf: `http://www.w3.org/2005/02/xpath-functions`. Es ist daher nicht notwendig, ein Präfix bei XSLT 2.0 Funktionen anzugeben.

Unterstützung von XPath 2.0-Funktionen
Nachfolgend sehen Sie eine Liste vordefinierter XPath 2.0-Funktionen mit eingeschränkter Unterstützung.

Funktion	Eingeschränkte Unterstützung
`fn:lower-case`	Gilt nur für den lateinischen Zeichensatz.
`fn:normalize-unicode`	Nicht unterstützt.
`fn:upper-case`	Gilt nur für den lateinischen Zeichensatz.

Unterstützung von XSLT 2.0-Funktionen
Nachfolgend sehen Sie ein Liste von integrierten XSLT 2.0-Funktionen mit eingeschränkter Unterstützung.

Funktion	Eingeschränkte Unterstützung
`format-date`	Eingeschränkte Unterstützung wie bei `format-dateTime`
`format-dateTime`	Presentation Modifier und Formatting Tokens in den Variablen-Markern des Picture-Arguments werden nicht unterstützt die optionalen Argumente Language, Calendar und Country werden nicht unterstützt.
`format-time`	Eingeschränkte Unterstützung wie bei `format-dateTime`

Schema Awareness
Der Altova XSLT 2.0-Prozessor ist nicht schema-aware, d.h. nicht schemafähig. Daraus ergibt sich:

- Für type constructors werden nur die integrierten XML-Schema- und XPath-Datentypen unterstützt; benutzerdefinierte Typen werden nicht unterstützt.
- Die Validierung gegen ein Schema wird nicht unterstützt.

Datentypen in XPath 2.0
Wenn Sie einen XPath 2.0-Ausdruck für ein XML-Dokument evaluieren, das ein XML-Schema referenziert und gemäß diesem Schema gültig ist, müssen Sie Datentypen, die nicht durch eine Operation implizit in den benötigten Datentyp konvertiert werden, explizit konstruieren oder "casten". Im XPath 2.0 Datenmodell, das vom Altova XSLT 2.0-Prozessor verwendet wird, wird allen "**atomized"** Nodewerten aus dem XML-Dokument der Datentyp `xdt:untypedAtomic` zugewiesen. Der Typ `xdt:untypedAtomic` funktioniert gut mit impliziten Typkonvertierungen. Beispiel: Der Ausdruck `xdt:untypedAtomic("1") + 1` hat als Ergebnis den Wert 2, da der Wert `xdt:untypedAtomic` implizit vom Plus-Zeichen in `xs:double` geändert wird. Arithmetische Operatoren konvertieren Operanden implizit in `xs:double`.
Operatoren zum Vergleich von Werten konvertieren Operanden vor dem Vergleich in `xs:string`.

In einigen Fällen muss der benötigte Datentyp jedoch explizit konvertiert werden. Wenn Sie z.B. die zwei Elemente `startDate` und `endDate` haben, die im XML-Schema als Datentyp vom Typ `xs:date` definiert sind, ergibt die Verwendung des XPath 2.0-Ausdrucks `endDate -`

 © 2005 Altova GmbH

`startDate` einen Fehler. Wenn Sie allerdings `xs:date(endDate) - xs:date(startDate)` oder `(endDate cast as xs:date) - (startDate cast as xs:date)` verwenden, wird der Ausdruck korrekt zu einer Einzelsequenz vom Typ `xdt:dayTimeDuration` aufgelöst.

Whitespaces im XML-Dokument
Standardmäßig entfernt der Altova XSLT 2.0 Prozessor alle Whitespaces in Nodes, die nur Whitespaces enthalten, aus dem XML-Quelldokument. Beachten Sie, dass das Vorhandensein, bzw. Fehlen von Nodes, die nur Whitespaces enthalten, einen Einfluss auf den Wert hat, den die `position()`-Funktion zurückgibt.

Unterstützung von XSLT 2.0-Elementen
Nachfolgend sehen Sie eine Liste von XSLT 2.0-Elementen, die nur beschränkt unterstützt werden.

Elemente	Eingeschränkte Unterstützung
`xsl:number`	Nicht unterstützt.
`xsl:key`	Nicht unterstützt.
`xsl:strip-space`	Nicht unterstützt.
`xsl:preserve-space`	Nicht unterstützt.

13.19 Altova XQuery 1.0-Prozessor

Der Altova XQuery 1.0-Prozessor entspricht den Vorgaben des XQuery 1.0 Working Draft vom 11. Februar 2005. Dabei handelt es sich jedoch unter Umständen nicht um die aktuelle Version, wenn Sie den Prozessor verwenden. Stellen Sie sicher, dass Sie den Draft vom 11. Februar 2005 als Referenz verwenden, wenn Sie mit dem Altova XQuery 1.0-Prozessor arbeiten. Der XQuery-Standard überlässt dem Anwender die Entscheidung, wie viele Funktionen implementiert werden. Weiter unten finden Sie eine Liste, in der erklärt wird, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Kodierung
Bei der unterstützten Zeichencodierung handelt es sich um UTF-8.

Namespaces
Die folgenden Namespace URIs und ihre dazugehörigen Bindings sind vordefiniert.

Namespace-Name	Präfix	Namespace URI
XML-Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema-Instanz	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
XPath-Datentypen	`xdt:`	`http://www.w3.org/2005/02/xpath-datatypes`
Vordefinierte Funktionen	`fn:`	`http://www.w3.org/2005/02/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/02/xquery-local-functions`

Beachten Sie bitte die folgenden Punkte:

- Der Altova XQuery 1.0-Prozessor erkennt die oben angeführten Präfixe und verbindet diese automatisch mit den entsprechenden Namespaces.
- Da der oben angeführte Funktions-Namespace der Standard-Funktions-Namespace in XQuery ist, muss das Präfix `fn:` nicht verwendet werden, wenn vordefinierte Funktionen (z.B.: `string("Hello")`) aufgerufen werden. Das Präfix kann jedoch verwendet werden, ohne dass der Namespace im Query-Prolog deklariert werden muss (z.B.: `fn:string("Hello")`).
- Sie können den Standard-Funktions-Namespace ändern, indem Sie im Query-Prolog den `default function namespace` Ausdruck deklarieren.
- Wenn Sie Typen aus den Namespaces für XML-Schema und XPath-Datentypen verwenden, können Sie die Präfixe `xs:` bzw. `xdt:` verwenden, ohne die Namespaces explizit deklarieren und diese Präfixe im XQuery-Prolog an diese binden zu müssen (Beispiele: `xs:date` und `xdt:yearMonthDuration`.). Wenn Sie andere Präfixe verwenden möchten, müssen Sie diese im XQuery-Prolog explizit für den Namespace deklarieren. (Beispiel: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)

Bei Zuweisung eines falschen Namespaces wird eine Fehlermeldung ausgegeben. Beachten Sie allerdings, dass einige Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` sind definiert). Das Namespace-Präfix gibt an, welcher verwendet werden soll.

XML-Quelldokument und Validierung
XML-Dokumente, die bei der Ausführung eines XQuery-Dokuments mit dem Altova XQuery 1.0-Prozessor verwendet werden, müssen wohlgeformt sein. Sie müssen jedoch nicht einem XML-

© 2005 Altova GmbH

Schema ensprechend gültig sein. XML-Dokumente, die mit einem XML-Schema verknüpft sind, werden vor Ausführung des XQuery-Dokuments **nicht** gegen das Schema **validiert,** d.h. im XML-Infoset, das an den Prozessor übergeben wird, sind keine Validierungsinformationen (einschließlich Typ-Informationen) inkludiert. Nodes aus dem XML-Dokument werden zu `xdt:untypedAtomic` aufgelöst, wenn Sie atomisiert werden. Der Typ `xdt:untypedAtomic` wird von einigen Operatoren implizit in den erforderlichen Typ konvertiert. So hat z.B. der Ausdruck `xdt:untypedAtomic("1") + 1` als Ergebnis den Wert 2, da der Wert `xdt:untypedAtomic` implizit vom Plus-Zeichen in `xs:double` geändert wird. Arithmetische Operatoren konvertieren Operanden implizit in `xs:double`. Operatoren zum Vergleich von Werten konvertieren Operanden vor dem Vergleich in `xs:string`.

In einigen Fällen muss der benötigte Datentyp jedoch explizit durch Konstruktion oder "Casten" in den benötigten Datentyp konvertiert werden. Wenn Sie z.B. die zwei XML-Elemente `startDate` und `endDate` haben, die im XML-Schema als Datentypen vom Typ `xs:date` definiert sind, ergibt die Verwendung des XPath 2.0-Ausdrucks `endDate - startDate` einen Fehler. Wenn Sie allerdings `xs:date(endDate) - xs:date(startDate)` oder `(endDate cast as xs:date) - (startDate cast as xs:date)` verwenden, wird der Ausdruck korrekt zu einer Einzelsequenz vom Typ `xdt:dayTimeDuration` aufgelöst.

Die XQuery `validate`-Anweisung wird nicht unterstützt.

Schema-Importe und benutzerdefinierte Typen
Die `import schema`-Anweisung und benutzerdefinierte Typen werden nicht unterstützt.

Statische und dynamische Typüberprüfung
In der statischen Analysephase werden Aspekte der Abfrage wie Syntax, das Vorhandensein von externen Referenzen (z.B. für Module), ob aufgerufene Funktionen und Variablen definiert sind usw. überprüft. In der statischen Analysephase wird keine Typüberprüfung durchgeführt. Wird in dieser Phase ein Fehler gefunden, erscheint eine Fehlermeldung und die Ausführung wird angehalten.

Die dynamische Typüberprüfung erfolgt in Runtime während der eigentlichen Ausführung der Abfrage. Erfüllt ein Typ die Voraussetzungen für eine Operation nicht, wird ein Fehler ausgegeben. So gibt der Ausdruck `xs:string("1") + 1` z.B. einen Fehler zurück, weil an einem Operand vom Typ `xs:string` keine Addition ausgeführt werden kann.

Bibliotheksmodule
In Bibliotheksmodulen sind Funktionen und Variablen zur späteren Wiederverwendung gespeichert. Der Altova XQuery 1.0-Prozessor unterstützt Module, die in einer **einzigen externen XQuery-Datei** gespeichert sind. Eine solche Moduldatei muss in ihrem Prolog eine `module`-Deklaration enthalten, die eine Verknüpfung mit einem Target Namespace herstellt. Im Folgenden sehen Sie ein Beispiel für ein Modul:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns: webaddress() { "http://www.altova.com" };
```

Alle im Modul deklarierten Funktionen und Variablen gehören zu dem mit dem Modul verknüpften Namespace. Die Verwendung des Moduls erfolgt durch Import des Moduls in eine XQuery-Datei mittels der `import module`-Anweisung im Query-Prolog. Die `import module`-Anweisung importiert nur Funktionen und Variablen, die direkt in der Bibliotheksmoduldatei deklariert sind. Dies sieht folgendermaßen aus:

```
import module namespace modlib = "urn:module-library";
if    ($modlib:company = "Altova")
then   modlib:webaddress()
else   error("No match found.")
```

Externe Funktionen
Externe Funktionen - also Ausdrücke, die das Stichwort `external` wie im folgenden Beispiel gezeigt verwenden - werden nicht unterstützt:

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations
Die Standard-Collation ist die Unicode Codepoint Collation. Derzeit wird keine andere Collation unterstützt. Vergleiche wie die `fn:max` Funktion basieren auf dieser Collation.

Character normalization
Es wird kein "character normalization form" unterstützt.

Präzision von numerischen Typen

- Der `xs:integer` Datentyp ist arbitrary-precision, d.h. er kann für jede beliebige Anzahl von Stellen stehen.
- Der `xs:decimal` Datentyp hat maximal 20 Stellen nach dem Komma (bzw. Dezimalpunkt).
- Die Datentypen `xs:float` und `xs:double` sind begrenzt auf 15 Stellen.

Unterstützung von XQuery-Anweisungen
Nachfolgend sehen Sie eine Liste mit XQuery-Anweisungen mit eingeschränkter Unterstützung.

Anweisungen	Eingeschränkte Unterstützung
`import schema`	Nicht unterstützt.
`MUExtensions`	Nicht unterstützt. Ignoriert.
`Pragma`	Nicht unterstützt. Ignoriert.
`schema-attribute`	Nicht unterstützt.
`schema-element`	Nicht unterstützt.
`validate`	Nicht unterstützt.

Unterstützung von XQuery-Funktionen
Nachfolgend sehen Sie eine Liste von vordefinierten XQuery-Funktionen mit eingeschränkter Unterstützung.

Funktion	Eingeschränkte Unterstützung
`fn:lower-case`	Gilt nur für den lateinischen Zeichensatz.
`fn:normalize-unicode`	Nicht unterstützt.
`fn:upper-case`	Gilt nur für den lateinischen Zeichensatz.

© 2005 Altova GmbH

Altova XMLSpy 2005 Enterprise Edition

Referenz für Programmierer

Referenz für Programmierer

XMLSpy 2005 als Automation Server
XMLSpy 2005 ist ein Automation Server, d.h. es ist eine Applikation, die anderen Applikationen (den so genannten Automation Clients) programmierbare Objekte zur Verfügung stellt, sodass der Automation Client direkt auf die vom Automation Server verfügbar gemachten Objekte zugreifen kann. Dies ist deshalb von Vorteil für einen Automation Client, da er die Funktionalitäten von XMLSpy 2005 nützen kann. So kann ein Automation Client z.B. die XML-Validierungsfunktionen von XMLSpy 2005 verwenden. Entwickler können daher ihre Applikationen verbessern, indem Sie die vorhandenen Funktionalitäten von XMLSpy 2005 in ihren Applikationen nutzen.

Die programmierbaren Objekte von XMLSpy 2005 werden den Automation Clients über die **XMLSpy API**, eine COM API, zur Verfügung gestellt. In dieser Dokumentation finden Sie das Objektmodell der API und eine komplette Beschreibung der verfügbaren Objekte (siehe Abschnitt XMLSpyAPI). Außerdem kann die API auch mit einer Java-Implementierung verwendet werden. Im Kapitel Using the XMLSpy API with Java finden Sie eine Beschreibung dazu.

Anpassen von XMLSpy 2005 und Modifizieren der Funktionen
Sie können die Installation von XMLSpy 2005 durch Ändern und Hinzufügen von Funktionalitäten anpassen. Außerdem können Sie Formulare für Benutzereingaben erstellen und die Benutzeroberfläche so anpassen, dass sie bestimmte neue Menübefehle und Symbolleistenschaltflächen enthält. All diese Funktionen werden in Form von Skripts implementiert, die mit Objekten der XMLSpy API interagieren. Um diese Funktionen effizient umsetzen zu können, bietet Ihnen XMLSpy 2005 eine integrierte Skripting-Umgebung. Im Abschnitt Scripting finden Sie eine komplette Beschreibung der Verwendung der Skripting-Umgebung und der darin verfügbaren Funktionen.

Des Weiteren können Sie XMLSpy 2005 mit Hilfe von externen Skripts manipulieren. So könnten Sie z.B. ein Skript schreiben, das XMLSpy 2005 zu einem bestimmten Zeitpunkt öffnet, anschließend die XML-Datei in XMLSpy 2005 öffnet, die Datei validiert und ausdruckt. Diese externen Skripten würden sich zur Durchführung dieser Aufgaben ebenfalls der XMLSpy API bedienen. Eine Beschreibung der XMLSpy API finden Sie im Abschnitt XMLSpyAPI.

Erstellen von Plug-Ins für XMLSpy 2005
XMLSpy 2005 gestattet Ihnen, Ihre eigenen Plug-Ins zu erstellen und in XMLSpy 2005 zu integrieren. Sie können dies mit Hilfe der speziellen Plug-In-Schnittstelle von XMLSpy 2005 tun. Eine Beschreibung zur Erstellung von Plug-Ins finden Sie im Abschnitt XMLSpy Plug-ins.

Informationen zur Referenz für Programmierer
Die in dieser Referenz für Programmierer enthaltene Dokumentation besteht aus den folgenden Abschnitten:

- Release Notes, in der die Änderungen seit der letzten Version aufgelistet sind
- Scripting: eine Benutzerreferenz für die Skripting-Umgebung in XMLSpy 2005
- XMLSpy Plug-ins: eine Beschreibung zur Erstellung von Plug-Ins für XMLSpy 2005
- XMLSpy API: eine Referenz für die XMLSpy API
- Using XMLSpy API with Java: eine Referenz zur Verwendung der XMLSpy API in Verbindung mit Java

© 2005 Altova GmbH

1 Release Notes

For each release of the XMLSpy API, important changes since the previous release are listed below.

Automation Interface for XMLSpy 2005 - type library version 1.4
Changes since XMLSpy API ver 2004R4:

- Bug fixes.

Automation Interface for XMLSpy 2004R4 - type library version 1.4
This is the current release. Important changes since the previous release (XMLSpy API 2004R3) are as follows:

Authentic View

- AuthenticView, AuthenticRange and AuthenticDataTransfer interfaces replace `DocEdit`, `DocEditSelection` and `DocEditDataTransfer`.
- The `AuthenticView` and `AuthenticRange` interfaces now supersede the old `DocEdit` interface.
- The `DocEdit` interfaces have been renamed `OldAuthentic` and are now obsolete.
- The `DocEditSelection` interface has been renamed `AuthenticSelection` and is now obsolete.
- The `DocEditDataTransfer` interface has been renamed `AuthenticDataTransfer`.
- The functionality of now obsolete interfaces are still available. However, usage of the new interfaces is strongly recommended.

Improved events
All events are available as connection point events to allow for easy and flexible integration of external clients using VBA, C++, VBScript, JScript or Perl. New events supporting the document life cycle have been added. All events pass event context information as parameters. Many events support cancellation of the signaled operation. See the section Overview of Events for more details.

Extension of Document interface

- Supports generation of program code (C++, C# and Java) for schema definitions
- Supports generation of MSWord and HTML documentation for schema definitions
- Supports XSL-FO transformation
- Extension of standard properties for documents
- Direct access to data root element of XML documents

Data import and export

- Choice of database for database import.
- Import of hierarchical data, which was previously done using a hand-written SHAPE statement with ImportFromDatabase, now requires a schema file and the use of ImportFromSchema.

Automation Interface for XMLSpy 2004R3 - type library version 1.3
Important changes since XMLSpy API ver 5 rel 4:

Authentic View

- Completely new interfaces AuthenticView and AuthenticRange to work with Authentic View.

2 Scripting

XMLSpy 2005 provides a **Scripting Environment** within the XMLSpy 2005 IDE. The Scripting Environment can be used to:

- Graphically design Forms for user input and for output
- Create, manage, and store scripts for these Forms
- Create scripts for application Event Handlers and for Macros.

Scripts access and interact with objects in the XMLSpy API, which is a COM API. They thus allow you to modify and add functionality to your installation of XMLSpy 2005. Programming languages that can be used in the Scripting Environment are **JavaScript** and **VBScript**.

Forms
In the Scripting Environment, you build a Form graphically using a palette of Form Objects, such as text input fields and buttons. For example, you can create a Form to accept the input of an element name and to then remove all occurrences of that element from the active XML document. For such a Form, a function script can be associated with the text input field so as to take an input variable; and an event handler can be associated with a button to start execution of the functionality, which is available in the XMLSpy API. For details, see Creating a Form.

Global Declarations
In the Scripting Environment, a set of Global Declarations contains the variables and functions that will be used by Forms, Event Handlers, and Macros. The functions make use of the XMLSpy API to access XMLSpy 2005 functionality.

Event handling
Using the Scripting Environment, you can easily customize XMLSpy 2005 by writing scripts for a variety of available events. You can control events that occur both within Forms (Form events) and within the general XMLSpy 2005 interface (application events).

Scripting code for an event is executed immediately upon the triggering of an event. Most events have parameters which provide detailed information on the event. The return value from the script typically instructs the application about how to continue its own processing (for example, the application may not allow editing). For details, see Creating an Event Handler.

Macros
Macros are used to implement complex or repetitive tasks. Macros do not use either parameters or return values. In a Macro, it is possible to access all variables and functions declared in the global declarations and to display forms for user input. Please see Writing a Macro for a simple example of creating a Macro. Also see Calling macros for a description of the ways in which a Macro can be called.

© 2005 Altova GmbH

2.1 Overview

This overview of the Scripting Environment discusses the following:

- The relationship between scripting projects and XMLSpy 2005 projects
- How and when Forms, Event Handlers and Macros are executed within XMLSpy 2005
- How settings for the Scripting Environment are to be made (in XMLSpy 2005)

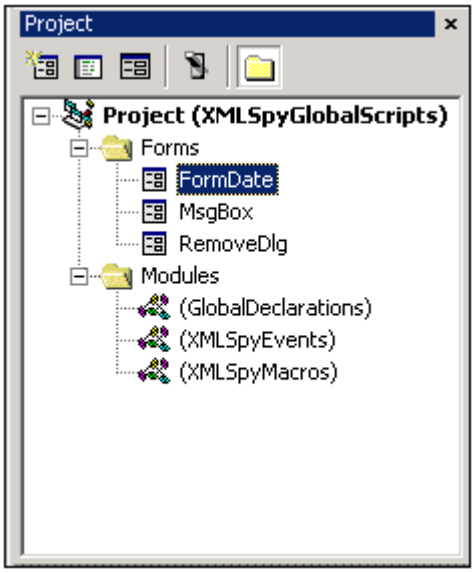
2.1.1 Scripting projects

Working with scripting projects

- In the Scripting Environment, all scripts and scripting information are stored in **scripting projects** (**`.prj` files**).
- After you have created and saved a scripting project, you must assign it to an XMLSpy 2005 project. The scripts in the scripting project can then be used when the XMLSpy 2005 project to which it has been assigned is opened in XMLSpy 2005.
- Only one scripting project can be assigned to an XMLSpy 2005 project at a time.
- You can assign a scripting project to more than one XMLSpy 2005 project.
- An XMLSpy 2005 project uses the scripts in the global scripting project and the scripts in the scripting project assigned to it.
- If you wish to make scripts applicable to the entire XMLSpy 2005 application, you must modify the global scripting project.
- The scripts in the global scripting project apply to the entire XMLSpy 2005 application even when no XMLSpy 2005 project is open.

The contents of a scripting project and how scripting projects are assigned to and unassigned from XMLSpy 2005 projects are described below.

Content of a scripting project
The Scripting Environment screenshot below shows the content of a typical scripting project.



The following points should be noted:

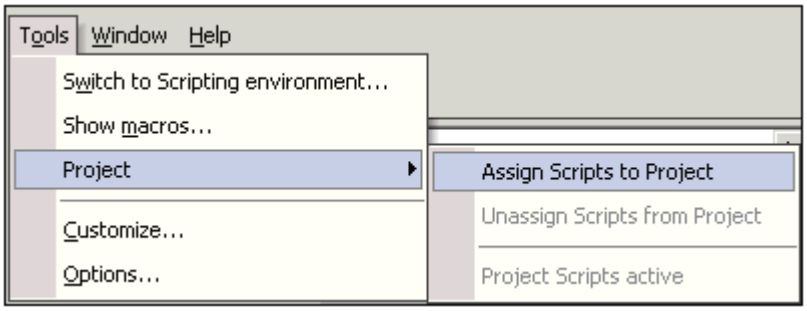
- A scripting project consists of two components: a **Forms** component and a **Modules**

component. The Forms component contains all the Forms defined for that scripting project. The Modules component comprises three modules: **Global Declarations**, **XMLSpy Events**, and **XMLSpy Macros**.The Global Declarations module contains definitions for variables and functions used in that scripting project. The XMLSpy Events module contains event handler scripts for that scripting project. The XMLSpy Macros module contains the macros that have been defined for that project.

- A scripting project, called `GlobalScripts.prj`, is installed with XMLSpy 2005. This scripting project contains default global declarations and template scripts for standard XMLSpy 2005 event handlers and macro tasks.
- Forms, events, and macros defined in the global scripting project can be accessed from within any XMLSpy 2005 project or XML file.
- In addition to the global scripting project, any number of additional scripting projects can be created within the Scripting Environment and saved.

Assigning scripting projects
A scripting project is assigned to an **XMLSpy 2005 project** from within the XMLSpy 2005 interface with the required XMLSpy 2005 project open.

To assign and unassign a scripting project to a project, use the **Tools | Project** submenu (see screenshot below). (These menu items are not enabled if there is no XMLSpy 2005 project active currently.)



To assign a scripting project, click **Assign Scripts to Project** and select the scripting project (`.prj` file) from the standard file-selection dialog that appears. The scripting project is loaded, and you are able to edit scripts in it if you switch to the Scripting Environment.

To unassign the current scripting project, select **Unassign Scripts from Project**. The scripting project is unassigned, and the Scripting Environment closes the scripting project.

The following points should be noted:

- For a given XMLSpy 2005 project, scripts from the scripting project specific to that XMLSpy 2005 project as well as scripts from the global scripting project can be accessed.
- Only **one** scripting project can be assigned to an XMLSpy 2005 project at a time.

Making a scripting project active
A scripting project is said to be active when it is the active project **in the Scripting Environment**. (More than one scripting project can be open in the Scripting Environment, but only one is active.)

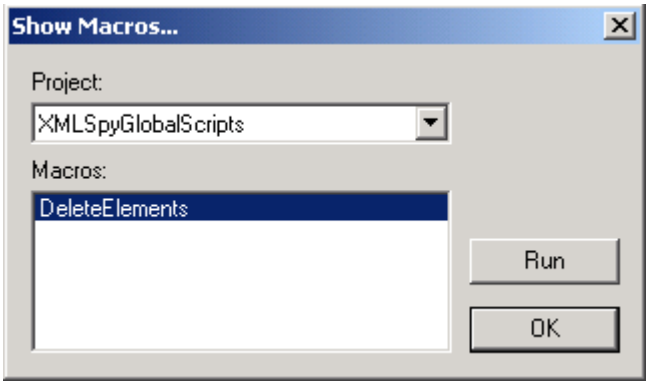
You can activate and deactivate the project-specific scripting project by toggling the **Project Scripts active** command on and off. Note that making the scripting project inactive **does not unassign** it; it merely deactivates the sripting project in the Scripting Environment.

 © 2005 Altova GmbH

2.1.2 Running Forms, Event Handlers, and Macros

Forms, Event Handlers, and Macros that have been defined in the Scripting Environment are all executed within the XMLSpy 2005 interface. However, the way they are called and executed is different for each and has a bearing on how you create your scripting projects.

- A Form is invoked by a call to it either within a function in the Global Declarations module or directly in a Macro.
- An Event Handler runs when the relevant event occurs in XMLSpy 2005. If an Event Handler for a single event is defined in both the global scripting project and the project-specific scripting project, then the Event Handler for the project-specific scripting project is executed first and that for the global scripting project immediately afterwards.
- A Macro is executed from within the XMLSpy 2005 interface by clicking **Tools | Show macros...**. This pops up the Show Macros dialog:



In the Project combo box, select either the global scripting project or the scripting project currently assigned to the XMLSpy 2005 project (only these two scripting projects are available as options). The macros defined in the scripting project you select appear in the Macros window. Select the macro you wish to run, and click **Run**.

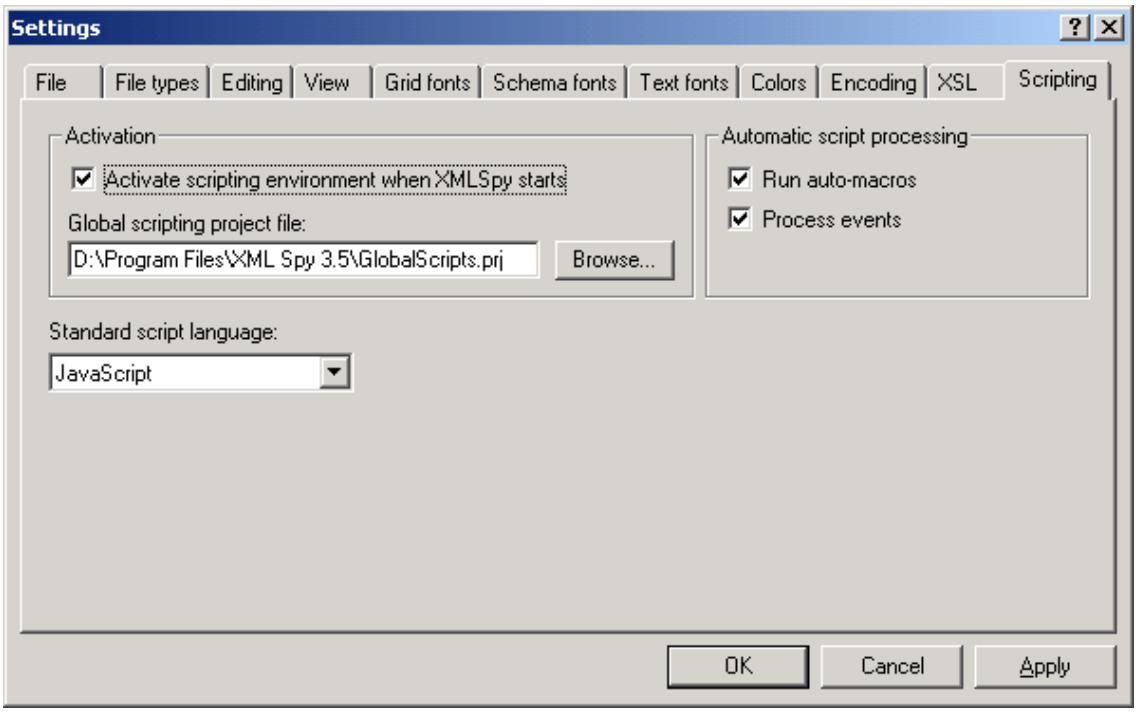
You can also customize your Tools menu to display a list of Macros (**Tools | Show Macros...**). A Macro in this list can subsequently be run by selecting it. See Calling macros for details.

Also see:

- Scripting Settings for information about configuring the XMLSpy 2005 interface for scripting.
- Calling macros for details about the various ways Macros can be called.
- The examples of how to create Forms, Global Declarations, Event Handlers, and Macros.

2.1.3 Scripting Settings

The screenshot below shows the Settings dialog for the Scripting Environment of XMLSpy 2005:



Activate scripting environment when XMLSpy 2005 starts
If this checkbox is selected, the Scripting Environment is opened as a separate window when XMLSpy 2005 starts. Otherwise, the Scripting Environment is not activated until the user selects **Tools | Switch to scripting environment** for the first time.

Global scripting project file
This entry specifies the global scripting project file. A sample global scripting project file is installed with your XMLSpy 2005 application. If this file does not exist when the Scripting Environment starts, a new scripting project with the specified name will be created at the specified location.

Run auto-macros
Enables or disables the execution of auto-macros. Currently Autorun from the global scripts is the only supported automatic macro. Autorun executes when the scripting environment starts.

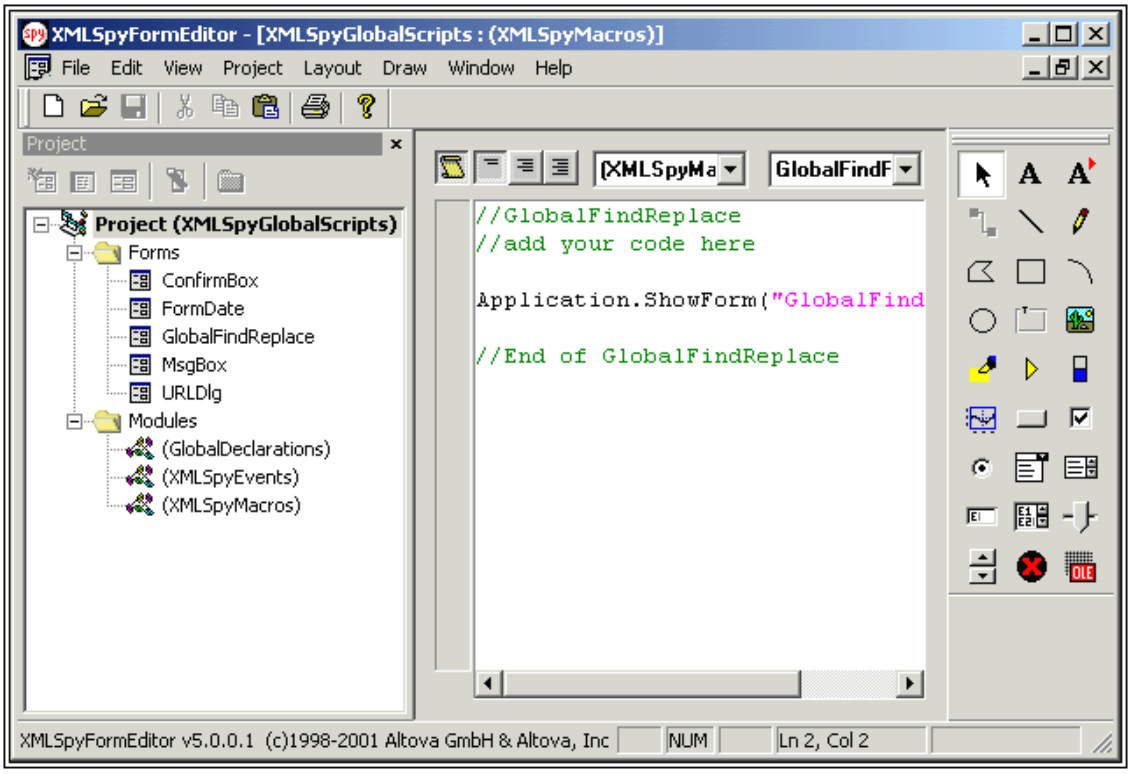
Process events
Enables or disables the execution of event handlers. If this check box is not ticked, then Event Handlers in the scripting projects will not be executed.

Standard script language
Every new scripting project created in XMLSpy 2005 will propose this scripting language as the default scripting language..

© 2005 Altova GmbH

2.2 The Scripting Environment GUI

The Scripting Environment GUI (shown below) is launched from XMLSpy 2005 by selecting **Tools | Switch to scripting environment...**. Alternatively, you can specify in XMLSpy 2005 (**Tools | Options | Scripting**) that the Scripting Environment be activated automatically each time XMLSpy 2005 starts.



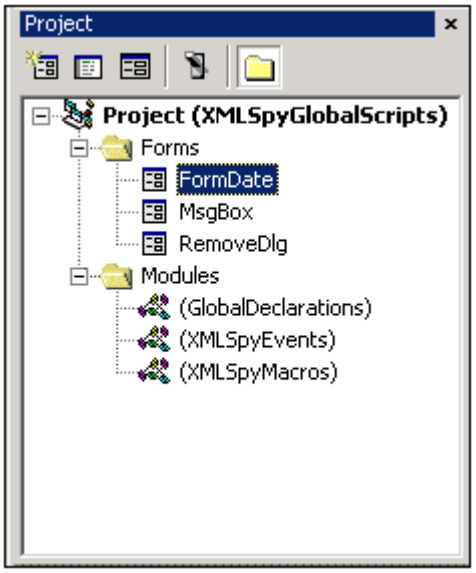
The Scripting Environment window has three parts: a Project Window, Main Window, and Form Object Bar. Each of these parts is described individually below. When a Form is active in the Main Window, the Scripting Environment becomes an effective graphical editor, called the **Form Editor**. The special features of the Form Editor are described separately below.

2.2.1 Project Window

Project Window

The Project Window displays all open scripting projects in a tree view. Only one scripting project can be active at a time. The title of the active project is displayed in bold.

To set a scripting project as the active project, right-click the scripting project and select **Set as active Project**. Closing a project removes it from the Project Window. If a project is not the active project, closing it automatically makes it the active project before it is closed—therefore, after it is closed no project is active.



Double-clicking a Form or Module in the Project Window, opens that Form or Module in the Main Window, where you can graphically design the Form or edit the script.

Icons
The following icons are in the Project Window's title bar:

New Form: Adds a new form to the Forms folder.

View Code: Displays the selected Form or the code of a Module. The code of the selected Form is displayed only if **Layout | Edit Script** has been toggled on for that Form. Otherwise the editable (if **Layout | Edit View** is selected) or non-editable (if neither **Layout | Edit Script** nor **Layout | Edit View** is selected) view of the Form is displayed. This icon is enabled when a Form or Module is selected.

View Form: Displays the selected Form. The editable view of the selected Form is displayed only if **Layout | Edit View** has been toggled on for that Form. Otherwise the code (if **Layout | Edit Script** is selected) or non-editable view (if neither **Layout | Edit View** nor **Layout | Edit Script** is selected) of the Form is displayed. This icon is enabled when a Form is selected.

Run Form: This icon is enabled when a Form is selecteds.

Toggle Folders: Toggles the Forms and Modules level of the tree on and off.

Context menus
Two context-sensitive popup menus are available within the Project Window. The first appears when you right-click the title of a project. The second appears when you right-click a Form or Module. These popup menus give you access to commands available in the **File** and **Project** menus.

© 2005 Altova GmbH

2.2.2 Main Window

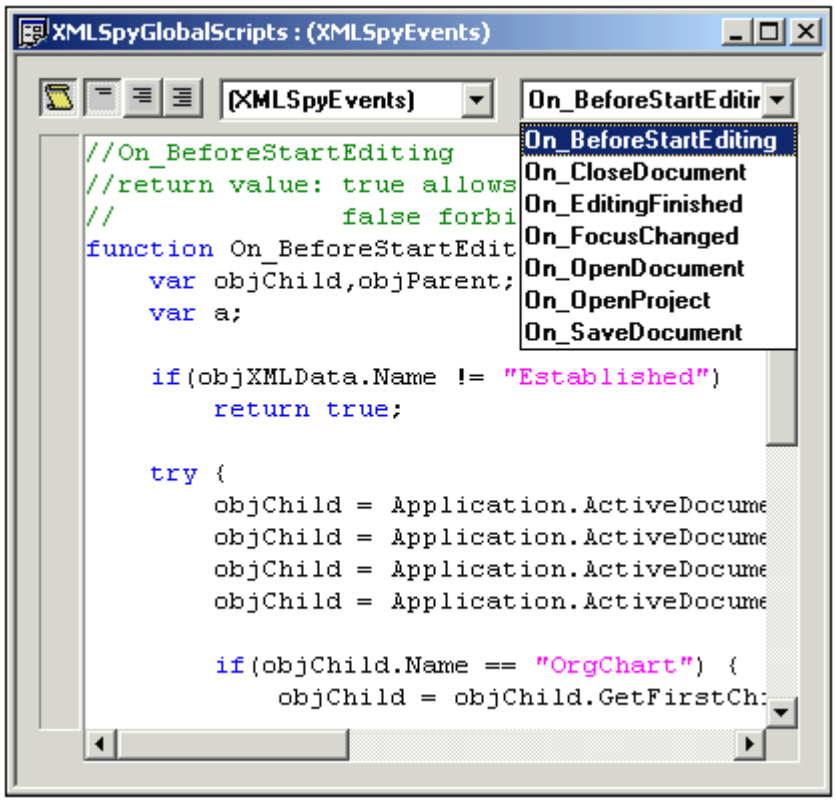
The Main Window is the area in the GUI where individual Forms and Modules are displayed in separate windows. It is in these windows that you design Forms and edit scripts.

Forms

- Each Form is displayed in a separate window.
- When the Scripting Environment GUI has a Form window active, it is called a Form Editor. For details of usage, see The Form Editor.
- A Form is displayed as the graphical dialog box presented to the user (Design View). If **Layout | Edit View** is toggled on for a given Form, then that Form is editable. Otherwise the Design View is non-editable.
- Additionally, if **Layout | Edit Script** has been toggled on for a Form, then the script underlying that Form is displayed in the Form window. Clicking the Close Script View icon switches back to the Design View (editable or non-editable according to whether **Layout | Edit View** is toggled on or off, respectively).
- Note that the **Layout | Edit View** and **Layout | Edit Script** switches are specified separately for each Form.

Modules

- Each Module (Global Declarations, (Application) Events, and Macros) opens in a separate window.
- Within a Module window, the combo box on the left indicates which module is selected (in the screenshot below, the combo box shows that the Events Module is the selected Module). To select a specific item (Global Declaration, Event-Handler, or Macro), click the required option in the drop-down menu of the combo box on the right.



Context menus

Right-clicking within the Script View of a Form or within a Module window pops up a context

menu that gives you access to features that help you with your scripting.

2.2.3 Form Object Bar

The Form Object Bar is a tool palette that helps you add controls to your Forms. Select a Form Object from the bar, then place the pointer at the desired location in your Form and draw a rectangle to specify the size of the object.

The most commonly used objects are described below:

Static Text: Adds text fields such as captions or field descriptions.

Button: Adds a button. It is possible to assign bitmaps to these buttons.

Check Box: Adds a check box, which enables Yes–No type elements.

Combo Box: Adds a combo box, which allows the user to select an option from a drop-down menu.

List Box: Adds a list box, which displays a list of items for selection.

Edit Box: Defines a single line of text.

Multi-edit Box: Defines multiple lines of text.

ActiveX: Allows the integration of an ActiveX control in the Form.

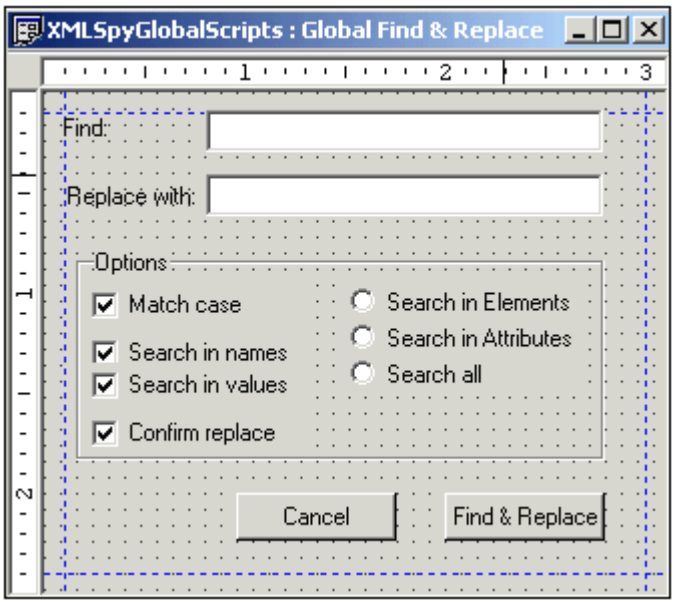
Setting Form Object properties and events
Each Form Object has its own set of properties and events. You can specify the settings of a Form Object in the Property Sheet for that object, as well as add some scripting code to the provided events. For details about how to do this, see The Form Editor.

 © 2005 Altova GmbH

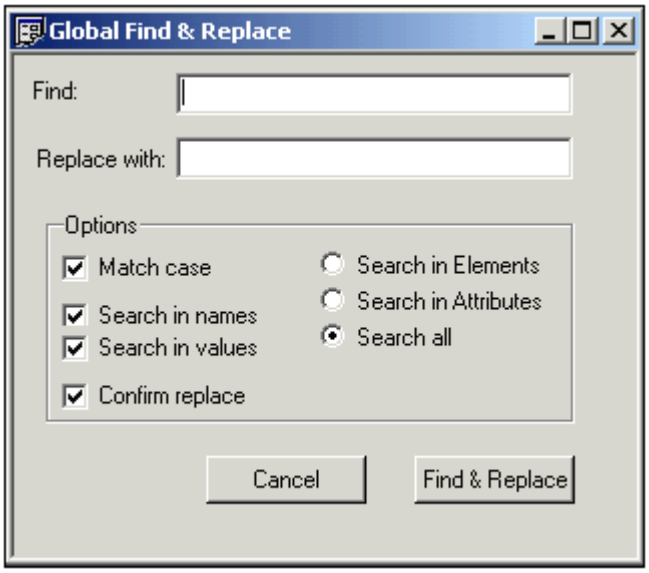
2.2.4 The Form Editor

The Form Editor is the Scripting Environment used for editing a Form. A Form can be opened by double-clicking that Form's icon in the Project Window or by right-clicking a Form icon and selecting either **View Code** or **View Object**.

Form views
Each Form opens in a separate window (shown below) that can be resized and minimized. The two views in which a Form is displayed are Design View and Script View.

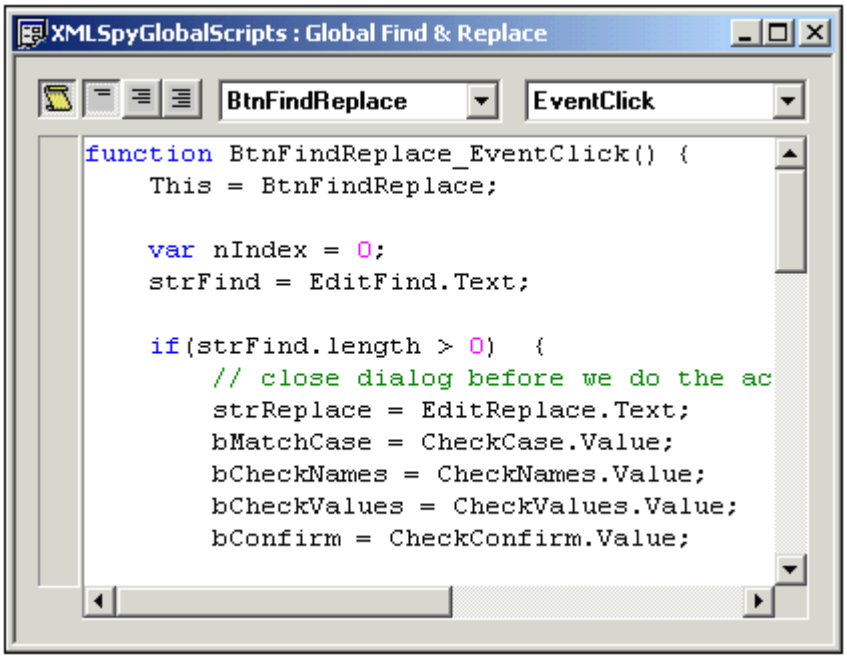


The Design View shown above is editable. In order to make a Design View editable, you must toggle on the menu item **Layout | Edit View**. If this menu item is toggled off, then the Design View is uneditable and the Form appears as it will to the user (see screenshot below).



Additionally, a Form can be displayed in a Script View (shown below) by toggling on the menu item **Layout | Edit Script**. Closing the Script View switches the view back to Design View

(editable or non-editable according to whether **Layout | Edit View** is toggled on or off, respectively).

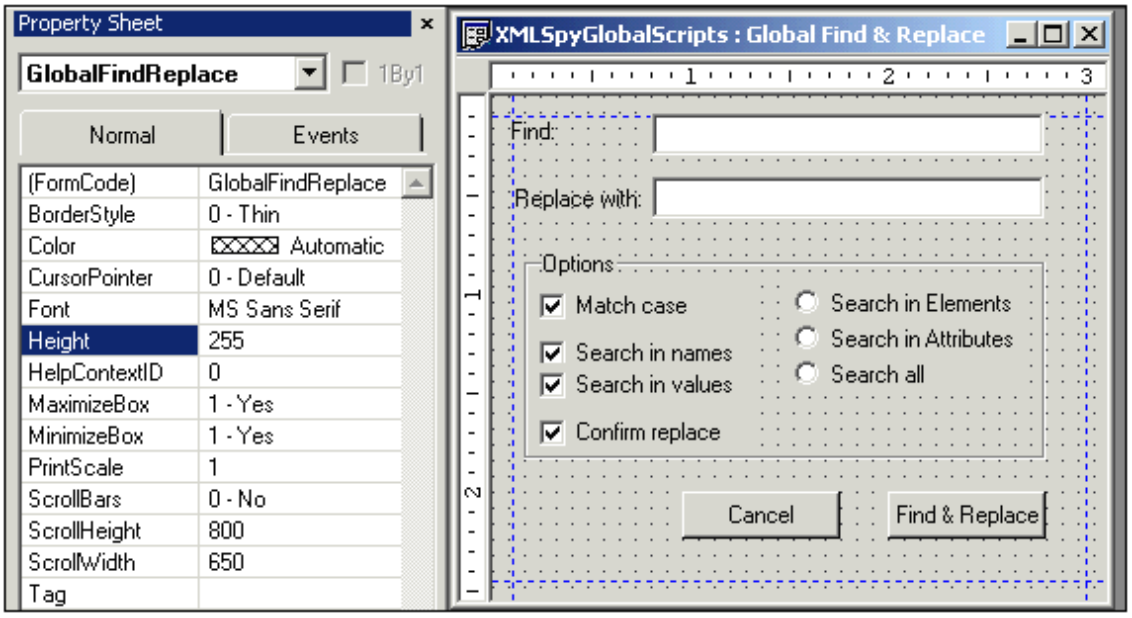


Design View
A Form must be created and edited in the editable Design View of the Form. To specify the settings of the workarea, use the menu item **Layout | Grid Settings...**.

In order to insert objects into the Form, use the Form Objects in the Form Object Bar. Build up the visual design of your Form by placing objects from the Form Object Bar at suitable locations within the Form outline. Specify visual refinements of the layout through the **Layout** menu items.

To specify the properties of the Form, either right-click in an empty area of the Form or select **Layout | Properties...**. This displays the Properties Sheet of the Form, where you can define the size of the Form, its font, buttons, and so on.

 © 2005 Altova GmbH

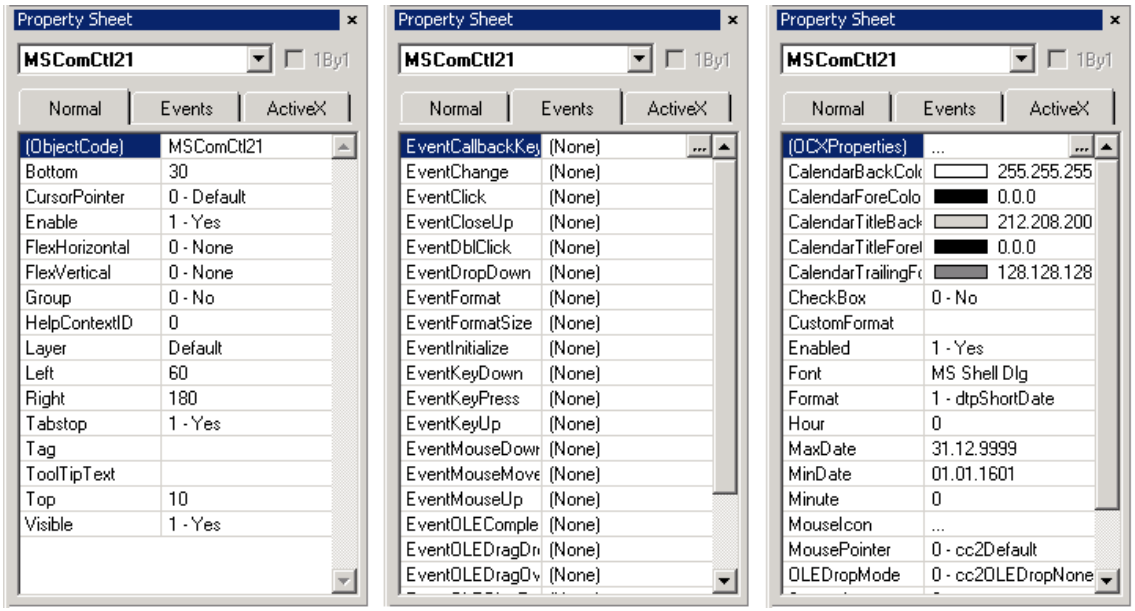


To switch to the Property Sheet of a particular object, either select that object in the design view or select the name of that object from the drop-down list of the combo box in the Property Sheet.

To close the Property Sheet, click the **Close** button.

The Property Sheet
The Property Sheet lists all properties and events of the selected Form or Form Object when the editable Design View of the Form is active. It enables you to modify properties and to add or modify code for properties and Form event handlers. The combo box at the top of the Property Sheet lists all the properties and events of the active Form.

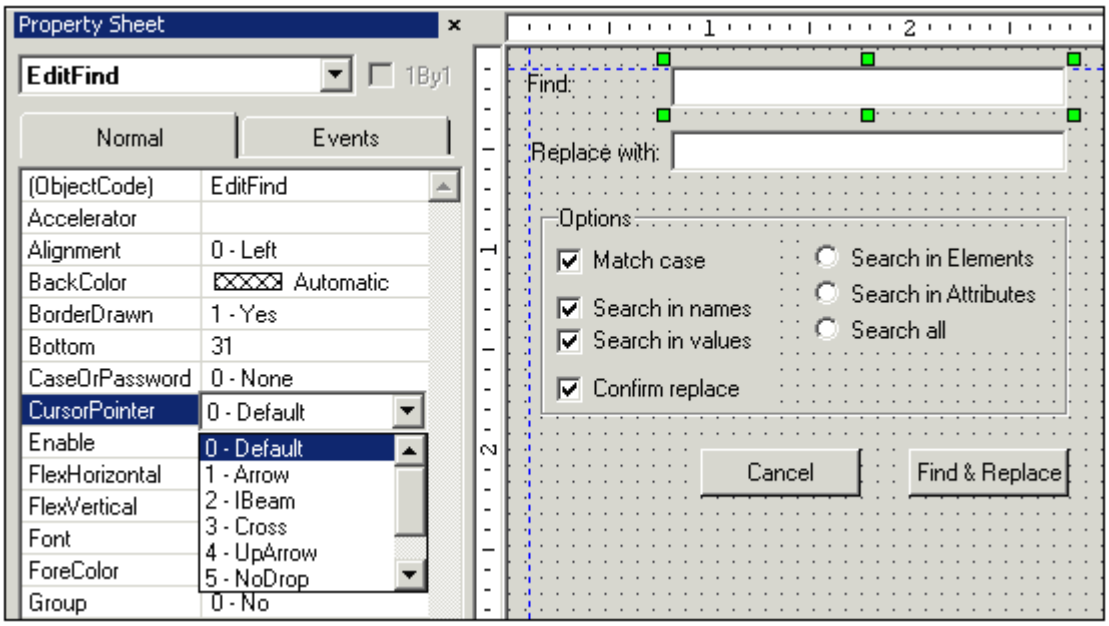
The Property Sheet has three tabs: Normal, Events, and ActiveX. The screenshots below show the three tabs of the Microsoft DatePicker ActiveX control. Note that different Form Objects have different properties and events, and not all objects have three tabs in the Property Sheet.



The **Normal tab** specifies the name of the object and properties relating to the object's appearance in the Form. The name is specified by the `ObjectCode` property (here `MScomCtl21`). If you write a script and need access to methods or properties of this object, you must use this name. For example:

```
var strTmp;
strTmp = "";
strTmp = MSComCtl21.Year + '-' + MSComCtl21.Month + '-' +
   MSComCtl21.Day;
```

If a property has enumerated values, then these become available in a drop-down menu when the property is selected (see screenshot below).



In the above screenshot, the `EditFind` object is selected. When the `CursorPointer` property is selected, a drop-down menu with the possible values for this property becomes available. This enables you to quickly and correctly select a suitable value for the property.

 © 2005 Altova GmbH

In the **Events and ActiveX tabs**, to enter code for a property, first select the property for which you want to enter code. A button with an ellipsis (. . .) appears at the right of the entry field. Click this button to open a Script Editor window for that code fragment.

Using layers in Form design
When you insert Form Objects while in the Design View of the Form, these objects are created by default on a layer titled "Default". You can add additional layers **below the Default layer** if you wish to create a Form with backgrounds objects. The objects in each layer are superimposed on the objects of underlying layers if there is an overlap. Layers below the Default layer can be moved relative to each other (i.e. up or down), but the Default layer remains the top layer. As a result, objects in the Default layer will be superimposed on all objects in lower layers. For details of using layers, see Layout.

Setting the tab sequence for data-input objects
When users input data in a Form, they commonly move from one input field to the next by pressing the Tab key. You can specify the tab sequence in the Objects Sheet (Tab Order) dialog. See Layout for details.

Also see:

- The Layout menu for commands used to layout a Form. There are a range of commands that help you to format
- Creating a Form for more details about creating a Form.

2.3 Using the Scripting Environment

The Forms, Event Handlers, and Macros you create in the Scripting Environment use scripts that interact with the XMLSpy API, which is a COM API. The variables and functions required for this interaction are defined in the **Global Declarations** module of the Scripting Environment. The object model of the XMLSpy API is documented in the XMLSpyAPI documentation.

Event Handler scripts can be written for standard XMLSpy 2005 events as well as for custom events. This allows you to add and modify event handling in your XMLSpy 2005 interface. See Creating an Event-handler for an example of how to work with Event Handlers in XMLSpy 2005.

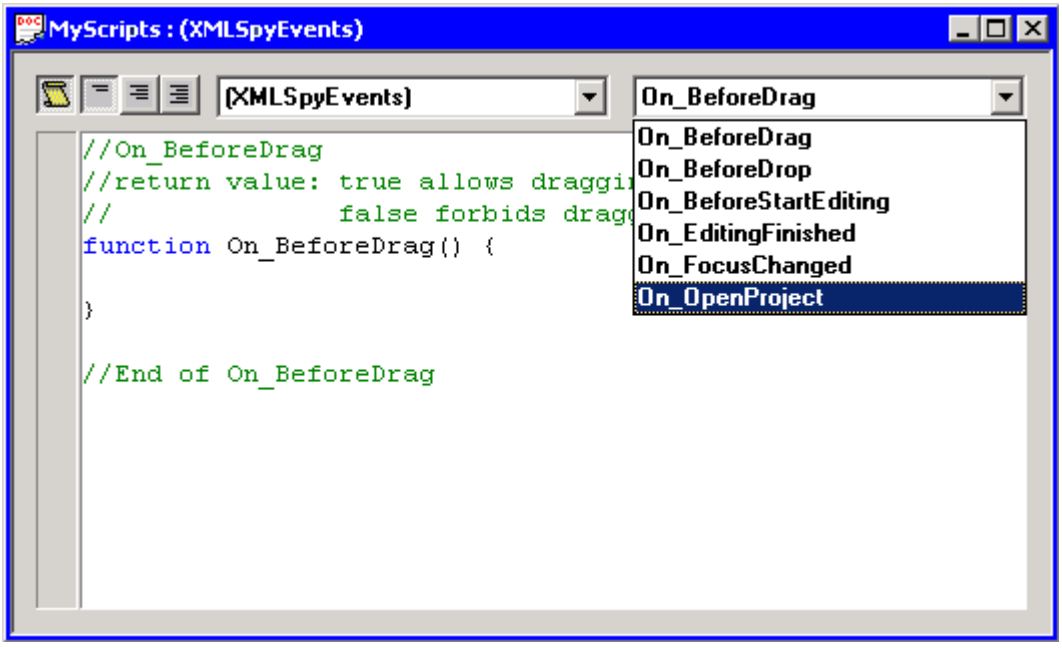
When you create a **Form**, you require variables to accept user input and functions that carry out the required functionality. Both variables and functions are defined in the Global Declarations. In addition, you need to reference some event handler to execute the Form in the desired way. Creating a Form describes the steps required to create a Form.

Macros automate repetitive or complex tasks. In the Scripting Environment, you can create a script that calls functions both XMLSpy 2005 functions as well as custom functions you define in the Global Declarations module. This flexibility provides you with powerful method of automating tasks within the XMLSpy 2005 IDE. How you create a Macro is described in Writing a Macro.

The subsections of this section explain, with examples, how each of the above is to be created. The section ends with a subsection explaining how Macros are called in XMLSpy 2005.

2.3.1 Creating an Event Handler

To create an Event Handler for application events (as opposed to Form events), double-click the XMLSpyEvents Module in the Project Window. This pops up a scripting window for event handlers in the Main Window.



The combo box on the right lists all the application events. To access a particular event-handler script, click the relevant event in the righ-hand combo box. This pops up the script which you can edit or a script outline you can fill in. If the event you want to write code for does not have an entry in the Event combo box, use the **Project | Add Function** command (or right-click in the

 © 2005 Altova GmbH

script window and select **Add Function**) to add an Event.

Note:

- Event Handlers need function headers with the correct spelling of the event name. Otherwise the Event Handler will not get called.
- It is possible to define local variables and helper functions within macros and event handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm("MsgBox");
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal = "On_BeforeStartEditing()";
  Helper();
}
```

- While a Macro is executed, Event Handlers from XMLSpy 2005 are not processed.
- The best way to re-create a deleted event handler is to create a new scripting project and copy name and function definition from there.
- In order for events to be processed, the Process Events options must be toggled on in the Settings dialog of XMLSpy 2005. See Scripting Settings for details.
- Also see Programming points to note.

Example
The screenshot above shows the XMLSpyEvents module with a list of predefined events. The `On_OpenProject` allows you to add a script that will be executed each time XMLSpy 2005 opens a project. The example script below sequentially opens all XML files located in the XML folder of the project and validates them. If the validation fails the script shows the validation error and stops. If a file passes the validity test, it will be closed and the next file will be opened.

Enter the following script for the `On_OpenProject()` event, and then save the scripting project.

```
function On_OpenProject()
{
 varbOK;
 varnIndex,nCount;
 varobjItems,objXMLFolder = null;

 objItems = Application.CurrentProject.RootItems;
 nCount = objItems.Count;

 // search for XML folder
 for(nIndex = 0;nIndex < nCount;nIndex++)  {
  var txtExtensions;
  txtExtensions = objItems.Item(nIndex).FileExtensions;

 if(txtExtensions.indexOf("xml") >= 0)     {
  objXMLFolder = objItems.Item(nIndex);
 break;
 }
}

// does XML folder exist?
if(objXMLFolder) {
 var objChild,objDoc;
```

```
nCount = objXMLFolder.ChildItems.Count;

// step through associated xml files
for(nIndex = 1;nIndex <= nCount;nIndex++) {
 objChild = objXMLFolder.ChildItems.Item(nIndex);

 try{
  objDoc = objChild.Open();

  // use JScript method to access out-parameters
  var strError = new Array(1);
  var nErrorPos = new Array(1);
  var objBadData = new Array(1);

  bOK = objDoc.IsValid(strError,nErrorPos,objBadData);

  if(!bOK){
   // if the validation fails, we should display the
   // message from XMLSpy
   // of course we have to create the form "MsgBox" and
   // define the global txtMessage variable
   //
   // txtMessage = Position:" + nErrorPos[0] + "\n" + // strError[0];
   // txtMessage += "\n\nXML:\n" + objBadData[0].Name + ", " +
   //   objBadData[0].TextValue;
   //
   // Application.ShowForm("MsgBox");

    break;
  }

  objDoc.Close(true);
  objDoc = null;
 }
 catch(Err) {
  // displaying the error description here is a good idea

  // txtMessage = Err.Description;
  // Application.ShowForm("MsgBox");

  break;
  }
 }
}
}
```

Testing the Event Handler
Switch to XMLSpy 2005, and open a project to see how the Open Project event is handled.

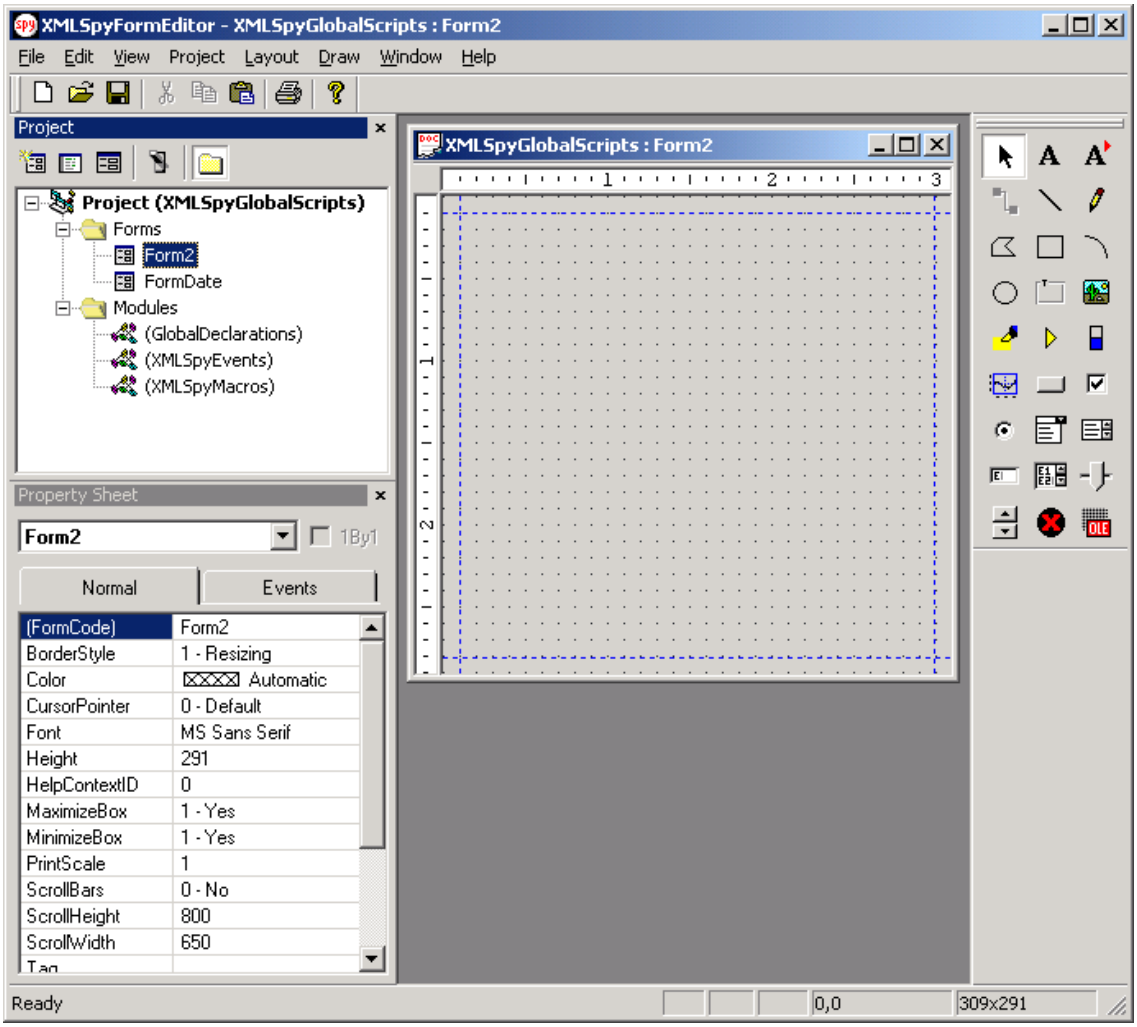
2.3.2 Creating a Form

A Form queries users for input (data and/or a choice), and executes the required action based on the input. In the Scripting Environment, you can graphically design a Form as well as create the scripts to perform the required action. The example below takes you through the steps needed to create a Form. In it we create a Form in which the user can type in the name of elements that are to be deleted from the active XML file in XMLSpy 2005.

Creating a new Form
To add a new Form to a scripting project, select **Project | Add Form** or click the **New Form** button in the toolbar of the Project Window. The new Form is added immediately. It appears as a separate window (see screenshot below), and its title appears in the Project Window. Open the Property Sheet for the Form by either right-clicking within the Form Window or selecting

 © 2005 Altova GmbH

Layout | Properties....



Naming the Form
You name a Form by entering a name for it in the line `FormCode` of the Property Sheet . For our example, enter `RemoveDlg` as the name of the Form. After entering the name, press Enter. The name of the Form now also appears in the title box of the Propery ySheet.

Inserting a Form Object
We need an edit box in which the user can type the names of the elements to be removed from the XML file. Select the edit box icon from the Form Objects Bar on the right side of the Main Window. Then, in the Design View of the `RemoveDlg` Form, draw, with the mouse, a rectangle for the edit box. This rectangle sets out the position and dimensions of the edit box. The edit box is created in the Design View, and (if **Layout | Properties...** is toggled on) the Property Sheet for the edit box is displayed. (To display the Property Sheet for the edit box, either select the edit box and toggle on **Layout | Properties...** or right-click the edit box.)

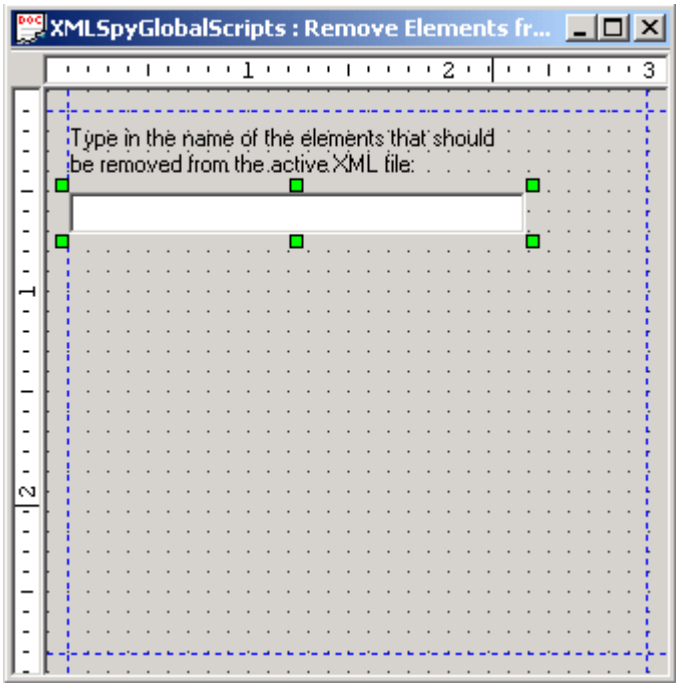
In the `(ObjectCode)` line of the Property Sheet, type in `EditElements` as the name of the edit box.

Position and dimension of a Form Object
The position and dimensions of the edit box are specified in the Property Sheet in terms of the

Left, Right, Top, and Bottom properties. The Left and Right properties take values that are the distances (in pixels) of the left and right borders of the edit box, respectively, from the left border of the Form. The Top and Bottom properties take values that are the distances (in pixels) of the top and bottom borders of the edit box, respectively, from the top border of the Form.

Inserting text in the Form

Add a caption for the edit box with the static text icon from the Form Objects Bar. The Design View of the Form should look something like this:



Inserting buttons

We now need two buttons in our form: **Delete** and **Cancel**. To insert these, do the following:

1. Select the button icon and then draw the two buttons in the Design View.
2. Name each button in the `(ObjectCode)` line of the respective Property Sheets: `BtnDelete` and `BtnCancel`.
3. Specify the Button text (**Delete** and **Cancel**, respectively) either by editing the button text directly in the Design View or by editing the value of the `Text` property in the Property Sheet.

Writing the required scripts

After the visual design of the Form is complete, we need to associate the Form Objects with suitable scripts. In this case, we need two types of scripts:

- A function that loops through the XML file, deleting all `XMLData` objects with a given name (specified by the user in the edit box)
- Event handlers that are executed when the two buttons are clicked

Creating the function

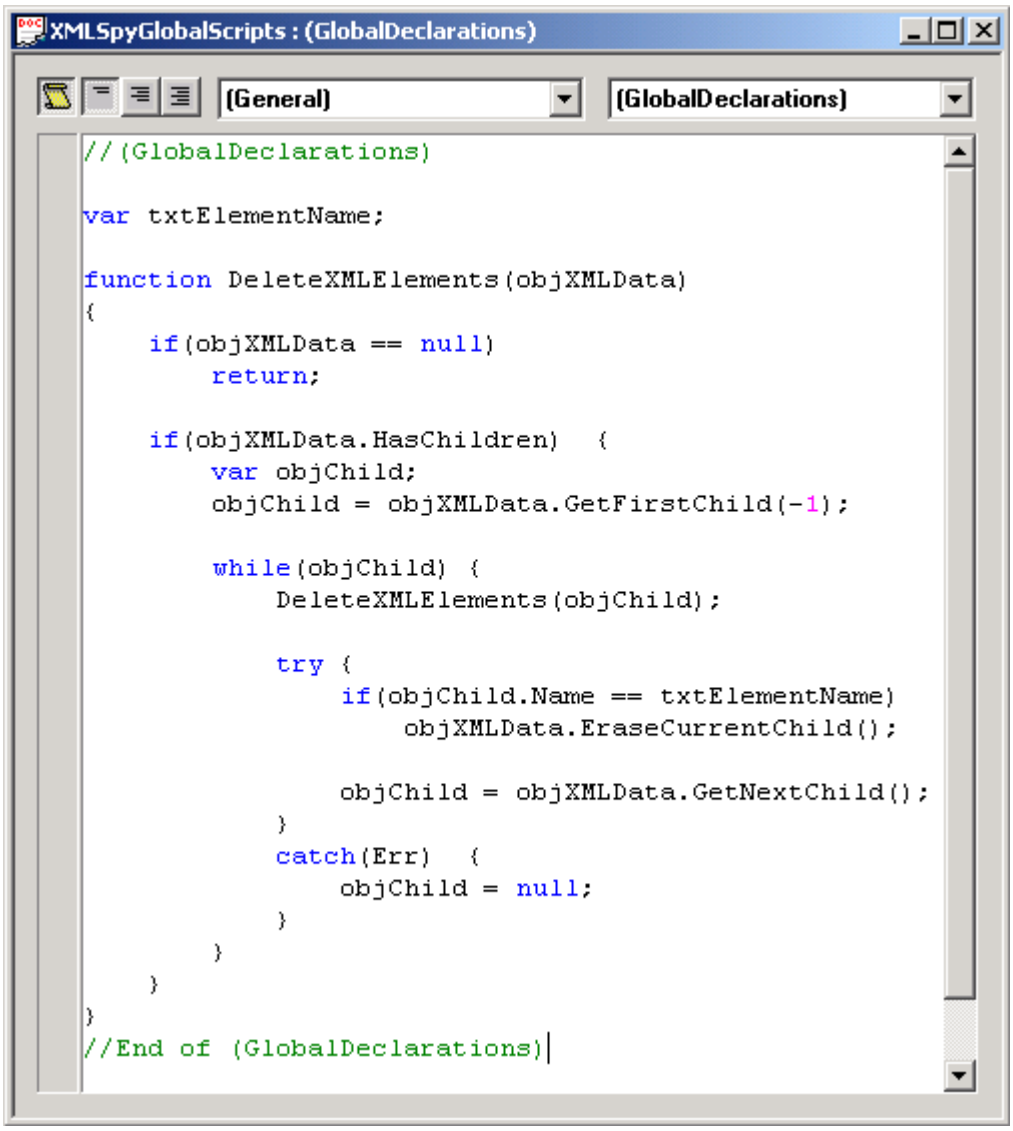
The function that deletes the `XMLData` objects must be defined in the Global Declarations module of the scripting project. Open the Global Declarations window by double-clicking the `GlobalDeclarations` module in the Project Window. Type the following code into the Global Declarations window:

```
var txtElementName;
```

 © 2005 Altova GmbH

```
function DeleteXMLElements(objXMLData)
{
  if(objXMLData == null)
    return;

  if(objXMLData.HasChildren){
    var objChild;
    objChild = objXMLData.GetFirstChild(-1);

    while(objChild) {
     DeleteXMLElements(objChild);

     try{
       if(objChild.Name == txtElementName)
       objXMLData.EraseCurrentChild();

       objChild = objXMLData.GetNextChild();
     }
     catch(Err) {
       objChild = null;
     }
   }
  }
}
```

The Global Declarations window should look like this:



Creating the button events

- Select the `BtnDelete` button and click the **Events** tab in the Property Sheet.
- Select the `EventClick` property and click the ellipsis button at the right. A Script Editor window pops up with the predefined function header of the click-event. This function will be executed every time the user clicks the **Delete** button. Add the following code to the event:

```
if(EditElements.Text != "")      {
            txtElementName = EditElements.Text;
            DeleteXMLElements(Application.ActiveDocument.RootElement);
}
```

Summary of Form creation

The Form we have created using the steps above does the following:

- The **Delete** button sets the global variable `txtElementName` to the string that the user enters in the edit box (`EditElements` is the name we have assigned the edit box).
- The global variable `txtElementName` calls the function `DeleteXMLElements()` with

 © 2005 Altova GmbH

the root element of the active XML document. (The function `DeleteXMLElements()` is the one we have defined in the Global Declarations module.)

- The `DeleteXMLElements()` function uses the `XMLData` object of the XMLSpy API.

Running the Form

To run the Form, create a Macro as follows:

1. Open the Macros module (by double-clicking `XMLSpyMacros` in the Project Window).
2. Create a new Macro called `DeleteElements` and enter thefollowing code:

```
var a;
if(Application.ActiveDocument != null)
            a = Application.ShowForm("RemoveDlg");
```

3. Run the Macro by selecting the `DeleteElements` in the Show Macros dialog, which you open by clicking **Tools | Show macros...** in the XMLSpy 2005 GUI.

Note:

- A Form can display another Form with the function `Application.ShowForm()`.

2.3.3 Writing a Macro

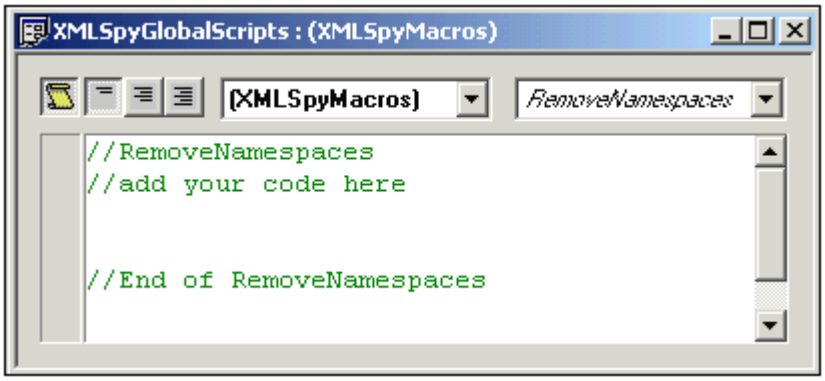
A Macro automates a repetitive or complex task. In this section, we write a Macro that removes all namespace prefixes from the active document.

Creating a new Macro

1. To create a new Macro, right-click `XMLSpyMacros` in the Project Window and select **Add Function** from the context menu.
2. In the New Function dialog that appears, enter `RemoveNamespaces` for the name of the Macro.

The Scripting Environment creates a new Macro with this name and adds it to the `XMLSpyMacros` module.

3. Double-click the `XMLSpyMacros` module in the Project Window to open the Macros window.
4. Select the `RemoveNamespaces` Macro.



5. Enter the code below in the Macro window:

```
if(Application.ActiveDocument != null)  {
            RemoveAllNamespaces(Application.ActiveDocument.RootElement
);
            Application.ActiveDocument.UpdateViews();
}
```

Note: Macros do not support parameters or return values, so no function header should exist in the macro implementation.

Creating the function used in the Macro

We now have to write the `RemoveNamespaces` function. Creating it in the Global Declarations module makes it accessible to all Macros, Event Handlers, and Forms.

1. Activate the Global Declarations module by double-clicking it in the Project Window.
2. Append the following code for the `RemoveNamespaces()` function to code for any previously defined global variables or functions:

```
function RemoveAllNamespaces(objXMLData)
{
      if(objXMLData == null)
    return;

  if(objXMLData.HasChildren)     {
    var objChild;

    // spyXMLDataElement := 4
    objChild = objXMLData.GetFirstChild(4);

    while(objChild) {
    RemoveAllNamespaces(objChild);

          try{
             var nPos,txtName;
             txtName = objChild.Name;

             if((nPos = txtName.indexOf(":")) >= 0)  {
                objChild.Name = txtName.substring(nPos+1);
             }

    objChild = objXMLData.GetNextChild();
          }
          catch(Err)             {
             objChild = null;
          }
      }
    }
  }
```

This completes the creation of the new macro.

Note:

- It is possible to define local variables and helper functions within macros and event handlers. Example:

```
//return value: true allows editing
//return value: false disallows editing
var txtLocal;
function Helper()
{
  txtMessage = txtLocal;
  Application.ShowForm("MsgBox");
```

 © 2005 Altova GmbH

```
}
function On_BeforeStartEditing(objXMLData)
{
  txtLocal = "On_BeforeStartEditing()";
  Helper();
}
```

- Recursive functions are supported. The function `DeleteXMLElements()` in Creating a Form, for example, calls itself recursively.

Running the Macro
Run the Macro by selecting the `DeleteElements` in the Show Macros dialog, which you open by clicking **Tools | Show macros...** in the XMLSpy 2005 GUI. See Calling macros.

Note: While a Macro is executed, Event Handlers from XMLSpy 2005 are not processed.

Modifying the Macro
This Macro can be easily modified to rename namespaces rather than remove them. You would need to carry out the following steps:

1. Design a Form (see Creating a Form) in which the user can specify the old and the new namespace names.
2. In the Macro code, change the `try-catch` block of the `RemoveAllNamespaces()` function to something like this:

```
try{
 var nPos,txtName;
 txtName = objChild.Name;

 if((nPos = txtName.indexOf(":")) >= 0){
  var txtOld;
  txtOld = txtName.substring(0,nPos);

  if(txtOld == txtOldNamespace)
   objChild.Name = txtNewNamespace + ":" + txtName.substring(nPos+1);
 }

 objChild = objXMLData.GetNextChild();
}
catch(Err){
 objChild = null;
}
```

This code assumes that `txtOldNamespace` and `txtNewNamespace` are declared as global variables and are set with the proper values.

2.3.4 Calling macros

There are two ways to call or run a Macro:

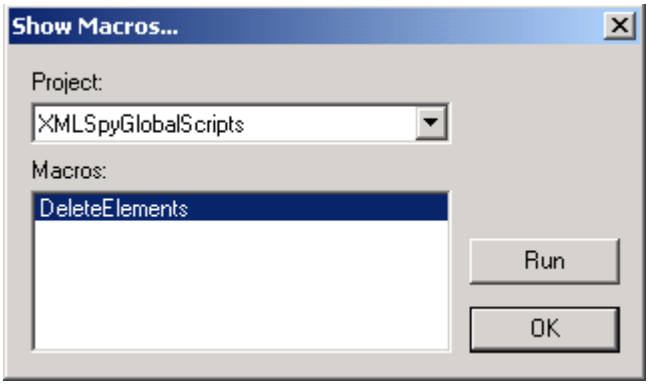
- With the **Show Macros...** dialog of XMLSpy 2005
- By creating and using a menu item (or Run command) for the Macro **in the Tools menu** of XMLSpy 2005
- By creating and using a toolbar button for a macro

Note: While a Macro is executed, Event Handlers from XMLSpy 2005 are not processed.

Using the Show Macros... dialog

1. In XMLSpy 2005, click **Tools | Show macros...**. This pops up the Show Macros... dialog, which displays all macros defined in the global scripting project and in the

scripting project assigned to the current XMLSpy 2005 project. In the Project combo box, select whether the Macros in the global scripting project or the Macros in the assigned scripting project are to be displayed.



2. Select the Macro you wish to run.
3. Click the **Run** button. Before the Macro is executed, the Show Macros... dialog is closed.

Creating a menu item in the Tools menu
The XMLSpy 2005 API includes a function to add Macros as menu items to the **Tools** menu. This function can be used to add one or more Macros to the **Tools | Show macros...** list of Macros. To do this carry out the following steps:

1. To add a menu item, call the XMLSpy API function `AddMacroMenuItem()`.
2. To reset the Tools menu call `ClearMacroMenu()`. This removes all previously added menu items

The best way to call these two functions is with the `Autorun` macro of the global scripting project or the `On_OpenProject` event.

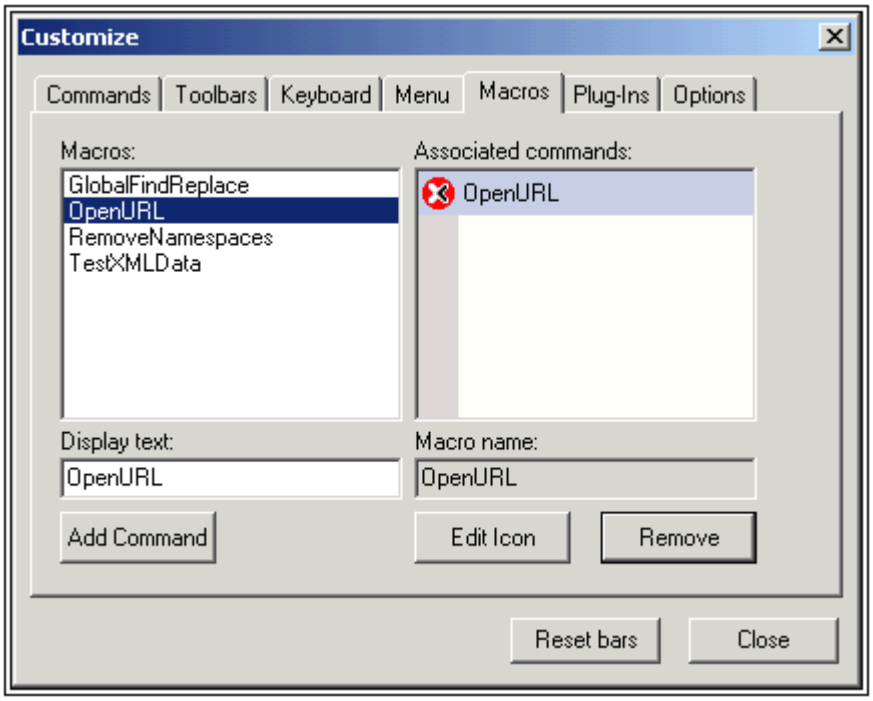
Example:

```
Application.AddMacroMenuItem("DeleteElements","Delete Elements
   Dialog");
```

- The first parameter (`DeleteElements` in the example) is the name of the Macro. If you run the Macro and there is an open project having scripts associated with it, XMLSpy 2005 searches for the Macro in the project scripts first. If there are no project scripts, or if XMLSpy 2005 cannot find the macro, then it looks for the Macro in the global scripts.
- The second parameter (`Delete Elements Dialog`)is the display text for the menu item.

Using a toolbar icon for a macro
You can create an icon in the toolbar, which, when it is clicked, will run a Macro. To do this, click **Tools | Customize | Macros**.

© 2005 Altova GmbH



Now do the following:

1. In the Macros pane, select the required Macro.
2. In the **Display text** input field enter the name of the icon. This name will appear when the cursor is placed over the icon when it is in the toolbar.
3. Click Add Command to add it to the list of commands.
4. Select the command and click Edit Icon to create a new icon.
5. Drag the finished icon from the Associated commands pane and drop it on to the toolbar when the cursor changes from an arrow to an I-beam.
6. Click Reset bars for this Macro icon and command to be displayed in the **Tools | Show Macros...** dialog.

2.3.5 Programming points to note

The following programming points should be noted:

- Out-parameters from methods of the XMLSpy API require special variables in JavaScript. Given below are some examples.

```
// use JavaScript method to access out-parameters
var strError = new Array(1);
var nErrorPos = new Array(1);
var objBadData = new Array(1);
bOK = objDoc.IsValid(strError,nErrorPos,objBadData);
```

2.4 Menus

2.4.1 File

New Project
Creates a new project. You are prompted for the scripting language of the new project, which can be either JavaScript or VBScript. The project is opened as the active project, and its title is diaplayed in bold in the Projects Window.

Open Project
Opens an existing project. The newly opened project becomes the active project. Only one project among all the open projects can be active at a time. A project is made the active project automatically when it is opened or if it is created as a new project. Otherwise, a project is made active by explicitly specifying this—right-click a project title and select **Set as Active Project**.

If only one instance of the project is open, then its title is displayed in bold. Multiple instances of a single project can be open, and in this case the title of the currently active instance or of the last instance to have been active is displayed in bold.

Close Project
Closes the selected project. If a project is not the active project, closing it automatically makes it the active project before it is closed—therefore, after it is closed no project is active.

Save Project
Saves the selected project.

Save Project As
Saves an unnamed open project with a name or a named open project under another name.

Printing
You can print a form or script by using the Print command. A Print Preview is also available. Note that to switch back from the Print Preview to the Scripting Environment, you should press the **Close** button. Do not close the Print Preview window; this will cause the entire Scripting Environment to close.

Switch to XMLSPY...
Minimizes the Scripting Environment and restores the XMLSpy 2005 window.

Exit
Closes the Scripting Environment window.

2.4.2 Edit

Undo and Redo
The Undo and Redo commands enable you to carry out multiple undo and redo commands.

Cut, Copy, Paste, Delete
To cut, copy and delete, select an object or text fragment, and select one of these commands. To paste, place the cursor at the desired location and select the Paste command.

© 2005 Altova GmbH

2.4.3 View

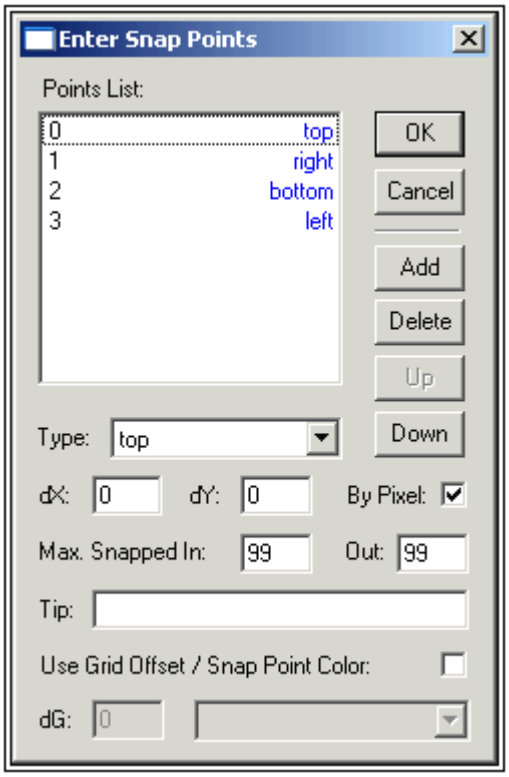
Toolbars
Toggles on/off a number of toolbars.

Project Bar
Toggles on/off the Project Bar.

Status Bar
Toggles on/off the Status Bar at the bottom of the Sripting Environment window.

Ruler Bars
Toggles on/off the ruler guides along the top and left margins of the page.

Snap Points
Toggles on/off the snap points of an object. Snap points are the points on an object that will snap to another object when that object is being moved. You can add and delete snap points by selecting the object, and, in the Property Sheet of the object, clicking the ellipsis in the value field of the AnchorSnaps property. This pops up the Enter Snap Ponts dialog, in which you can add snap points.



When the snap points are toggled on, they are visible and aid you to position objects accurately.

2.4.4 Project

Add Form
Enables you to add a Form to the current set of Forms in the project.

Add Function
When Events or Macros is selcted in the Project Window, you can add a Function to the list of

already available functions. Selecting this command pops up the following dialog:

Entering the name of the function and clicking OK adds a function with this name to the list of existing functions and opens a scriptig window for this function.

Remove Function
This command opens the Remove Function dialog, in which you enter the name of the Events or Macros functon to be removed.

Project Information
This command gives you the path to the current project file and information about whether the project was opened from within XMLSpy 2005 or not.

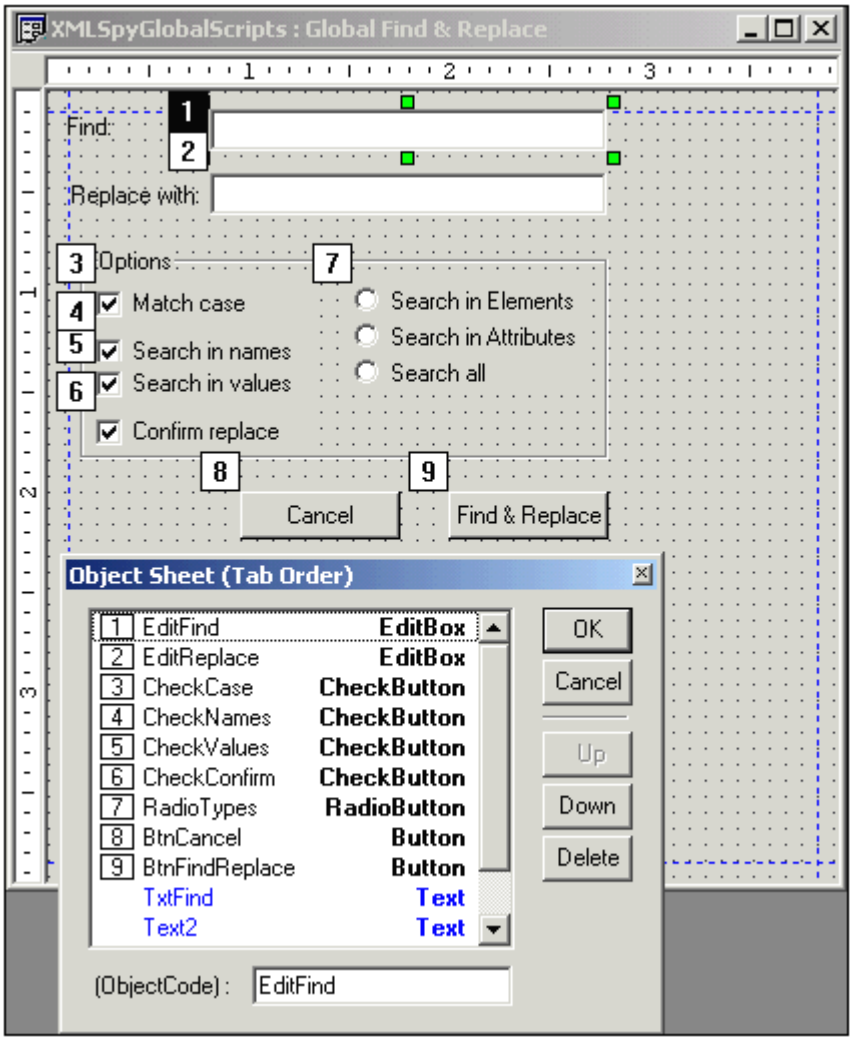
2.4.5 Layout

The Layout menu contains commands for Form layout and view controls. These commands are not relevant for Modules.

Spacing, positioning, and sizing of objects
When two or more Form Objects are selected (by pressing Shift while clicking the objects), they can be spaced and sized relative to each other as well as positioned relative to the borders of the Form. You can do this by using the **Align Objects**, **Space Evenly**, **Align in View**, and **Make Same Size** commands. Note that to use the **Space Evenly** command at least three objects have to be selected.

Objects
The **Objects...** command pops up the Object Sheet (Tab Order) window, in which you specify the tab sequence of the data-input fields. Data-input fields correspond to data-input Form Objects.

© 2005 Altova GmbH

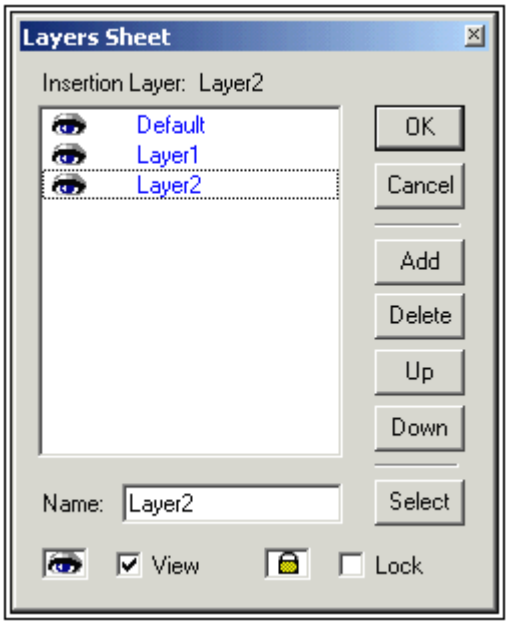


When you insert a Form Object in your design, the new object is appended to the appropriate section (numbered or non-numbered) in the Object Sheet list. The Object Sheet list has two parts, a numbered part for data-input objects and a non-numbered part for display objects. The number of an object in the numbered part indicates the position of that object in the run-time tab sequence. You can change the position of an object in the tab sequence by selecting it and pressing the **Up** or **Down** buttons as required. You can also delete an object by pressing the **Delete** button.

Note: The tab sequence applies to data-input objects only at run-time, i.e. when the Form is displayed for user input. In the Form Editor, pressing the Tab key first takes you through the display objects (in the order in which these are listed in the Object Sheet), and then through the data-input objects (in the tab sequence).

Layers
A Form is designed in a "Default" layer that is always the top layer. You can add additional layers below the Default layer, and move these lower layers relative to each other. The Default layer remains the top layer. Layers, therefore, allow you to add background effects to your Form.

To add and manipulate layers, select the **Layers...** command. This pops up the Layers Sheet.



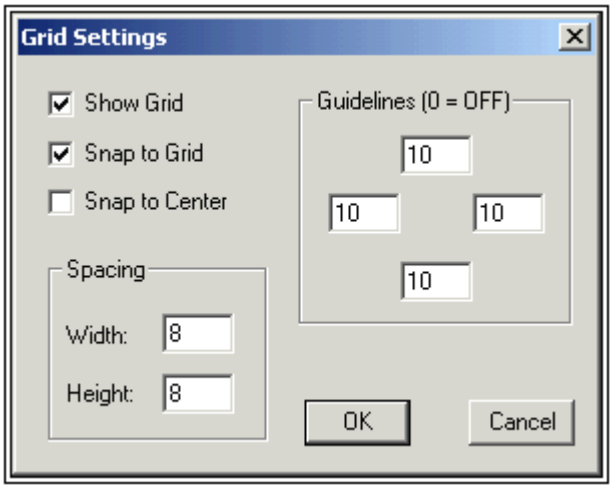
To add a new layer, click the **Add** button. You can rename a layer by selecting it and typing a new name into the **Name** input field. To delete a layer, select the layer to be deleted and click the **Delete** button. To move a layer up or down in the layer sequence, select the layer to move and press the **Up** or **Down** buttons as required. The layer sequence is ordered with the first layer in the Layers Sheet corresponding to the topmost layer of the Form.

The **Select** button selects, in the design view, all the objects of the currently selected layer. You can also turn the display of all objects in a layer on or off by using the **View** check box. Locking a layer, with the **Lock** check box, renders that layer uneditable and causes the Object Bar to be grayed out when that layer is active.

Properties
The **Properties...** command pops up the Property Sheet for the selected Form. A Property Sheet is available not only for each Form but also for each Form Object. You can modify properties of a Form or Form Object via its Property Sheet. How to use Property Sheets is described in detail in The Form Editor.

Grid Settings
The **Grid Settings...** command pops up the Grid Settings dialog, in which you can specify the appearance of the background grid for the design view.

© 2005 Altova GmbH



Spacing settings specify the horizontal and vertical distances between grid lines. Four guidelines are available (top, right, bottom, left), and you can specify the offset of each from the corresponding border of the Form. Grid Settings take effect with the next design view to be made active.

Edit View
This is a toggle setting that is specified individually for each Form. Checking the **Edit View** command causes the Form to be opened in editable design view or for it to switch from non-editable view to editable design view. Unchecking the **Edit View** command causes the Form to be opened in non-editable view or for it to switch from editable design view to non-editable view. Note that the setting is made for the active Form only; it does not apply to all Forms.

Edit Script
This is a toggle setting that is specified individually for each Form. Cecking the **Edit Script** command causes a separate script window to be displayed for that Form. Unchecking the command causes the script window to close.

2.4.6 Draw

Order
This is a Form Editor command and enables you to place the selected object in front of or behind other objects on the same layer. If you wish to arrange objects which are on different layers, then you should manipulate layers in the Layers Sheet (accessed with the **Layout | Layers** command).

Zoom
This is a Form Editor command menu and enables you to zoom in and out on an object or the Form as a whole.

Group
This is a Form Editor command menu which is enabled when two or more objects, or a group of objects, are selected. It allows you to group and ungroup two or more objects. This is useful if you wish to keep objects together as a group.

Rotate
Enables you to rotate certain objects. The **Free Rotate** command is a toggle command. It allows you to grab the object by one of its handles and rotate it about its center. You acn also **Rotate Right** by 90º. The **Unrotate** command functions like an Undo command.

2.4.7 Window

The Window menu enables you to manipulate the arrangement of open windows in the Main Window. You can cascade and tile windows, and arrange icons. You can also use this menu to select, from a list of all open windows, the window you wish to make the active window.

© 2005 Altova GmbH

3 XMLSpy Plugins

XMLSpy 2005 allows you to create your own IDE plug-ins and integrate them into XMLSpy 2005.

Use plug-ins to:

- Configure your version of XMLSpy 2005, add commands through menus, icons, buttons etc.
- React to events from XMLSpy 2005.
- Run your specific code within XMLSpy 2005 with access to the complete XMLSpy 2005 API

XMLSpy 2005 expects your plug-in to implement the IXMLSpyPlugIn interface. See ATL sample files for an example using C++. See the folder "xmlspy\Examples\XMLSpyPlugIn" of your XMLSpy 2005 installation for a sample using VisualBasic.

> If you are interested in developing a Plugin for XMLSPY using this API, please visit the Altova Partner Portal at **http://partners.altova.com**, or contact us via e-mail at mailto:partners@altova.com to learn more about the different partnership options!

3.1 Registration of IDE PlugIns

XMLSpy 2005 maintains a specific key in the Registry where it stores all registered IDE plug-ins:

HKEY_CURRENT_USER\Software\Altova\XML Spy\PlugIns

All values of this key are treated as references to registered plug-ins and must conform to the following format:

Value name:	ProgID of the plug-in
Value type:	must be REG_SZ
Value data:	CLSID of the component

Each time the application starts the values of the "PlugIns" key is scanned, and the registered plug-ins are loaded.

Register plug-in manually

To register a plug-in manually, use the "Customize" dialog box of the XMLSpy 2005 "Tools" menu. Use the "Add Plug-In..." button to specify the DLL that implements your plug-in. XMLSpy 2005 registers the DLL as a COM server and adds the corresponding entry in its "PlugIns" key.

If you experience problems with manual registration you can check if the CLSID of your plug-in is correctly registered in the "PlugIns" key. If this is not the case, the name of your plug-in DLL was probably not sufficiently unique. Use a different name or perform direct registration.

Register plug-in directly

A plug-in can be directly registered as an IDE plug-in by first registering the DLL and then adding the appropriate value to the "PlugIns" key of XMLSpy 2005 during plug-in setup for example. The new plug-in will be activated the next time XMLSpy 2005 is launched.

© 2005 Altova GmbH

3.2 Configuration XML

The IDE plug-in allows you to change the user interface (UI) of XMLSpy 2005. This is done by describing each separate modification using an XML data stream. The XML configuration is passed to XMLSpy 2005 using the GetUIModifications method of the IXMLSpyPlugIn interface.

The XML file containing the UI modifications for the IDE PlugIn, must have the following structure:

```
<ConfigurationData>
 <ImageFile>path To image file</ImageFile>
 <Modifications>
  <Modification>
   ...
  </Modification>
  ...
 </Modifications>
</ConfigurationData>
```

You can define icons or toolbar buttons for the new menu items which are added to the UI of XMLSpy 2005 by the plug-in. The path to the file containing the images is set using the ImageFile element. Each image must be 16 x 16 pixels using max. 256 colors. The image references must be arranged from left to right in a single (<ImageFile>...) line. The rightmost image index value, is zero.

The Modifications element can have any number of Modification child elements. Each Modification element defines a specific change to the standard UI of XMLSpy 2005. Starting with version 4.3, it is also possible to remove UI elements from XMLSpy 2005.

Structure of Modification elements
All Modification elements consist of the following two child elements:

```
<Modification>
 <Action>Type of action</Action>
 <UIElement Type="type of UI element">
 </UIElement>
</Modification>
```

Valid values for the Action element are:

Add - to add the following UI element to XMLSpy 2005
Hide - to hide the following UI element in XMLSpy 2005
Remove - to remove the UI element from the "Commands" list box, in the customize dialog

You can combine values of the Action element e.g. "Hide Remove"

The UIElement element describes any new, or existing UI element for XMLSpy 2005. Possible elements are currently: new toolbars, buttons, menus or menu items. The **type** attribute, defines which UI element is described by the XML element.

Common UIElement children
The ID and Name elements are valid for all different types of XML UIElement fragments. It is however possible, to ignore one of the values for a specific type of UIElement e.g. Name is ignored for a separator.

```
<ID></ID>
<Name></Name>
```

If UIElement describes an existing element of the UI, the value of the ID element is predefined

by XMLSpy 2005. Normally these ID values are not known to the public. If the XML fragment describes a new part of the UI, then the ID is arbitrary and the value should be less than 1000. The Name element sets the textual value. Existing UI elements can be identified just by name, for e.g. menus and menu items with associated sub menus. For new UI elements, the Name element sets the caption e.g. the title of a toolbar, or text for a menu item.

Toolbars and Menus
To define a toolbar its necessary to specify the ID and/or the name of the toolbar. An existing toolbar can be specified using only the name, or by the ID if it is known. To create a **new** toolbar both values must be set. The **type** attribute must be equal to "ToolBar".

```
<UIElement Type="ToolBar">
 <ID>1</ID>
 <Name>TestPlugIn</Name>
</UIElement>
```

To specify an XMLSpy 2005 menu you need two parameters:

- The ID of the menu bar which contains the menu. If no XML documents are open in the main window, the menu bar ID is 128. If one or more XML documents are open, the menu bar ID is 129.
- The menu name. Menus do not have an associated ID value. The following example defines the "Edit" menu of the menu bar which is active, when at least one XML document is open:

```
<UIElement Type="Menu">
 <ID>129</ID>
 <Name>Edit</Name>
</UIElement>
```

An additional element is used if you want to create a new menu. The Place element defines the position of the new menu in the menu bar:

```
<UIElement Type="Menu">
 <ID>129</ID>
 <Name>PlugIn Menu</Name>
 <Place>12</Place>
</UIElement>
```

A value of -1 for the Place element sets the new button or menu item at the end of the menu or toolbar.

Commands
If you add a new command, through a toolbar button or a menu item, the UIElement fragment can contain any of these sub elements:

```
<MacroName></MacroName>
<Info></Info>
<ImageID></ImageID>
```

If MacroName is specified, XMLSpy 2005 searches for a macro with the same name in the scripting environment and executes it each time this command is processed. The Info element contains a short description string which is displayed in the status bar, when the mouse pointer is over the associated command (button or menu item). ImageID defines the index of the icon the external image file. Please note that all icons are stored in one image file.

To define a toolbar button create an UIElement with this structure:

```
<UIElement Type="ToolBarItem">
 <!--don't reuse local IDs even the commands do the same-->
 <ID>5</ID>
```


© 2005 Altova GmbH

```
  <Name>Open file from repository...</Name>
  <!--Set Place To -1 If this is the first button To be inserted-->
  <Place>-1</Place>
  <ImageID>0</ImageID>
  <ToolBarID>1</ToolBarID>
  <!--instead of the toolbar ID the toolbar name could be used-->
  <ToolBarName>TestPlugIn</ToolBarName>
 </UIElement>
```

Additional elements to declare a toolbar button are Place, ToolBarID and ToolBarName. ToolBarID and ToolBarName are used to identify the toolbar which contains the new or existing button. The textual value of ToolBarName is case sensitive. The (UIElement) **type** attribute must equal "ToolBarItem".

To define a menu item, the elements MenuID, Place and Parent are available in addition to the standard elements used to declare a command. MenuID can be either 128 or 129. Please see "Toolbars and Menus" for more information on these values.

The Parent element is used to identify the **menu** where the new menu entry should be inserted. As sub menu items have no unique Windows ID, we need some other way to identify the parent of the menu item.

The value of the Parent element is a path to the menu item.
The text value of the Parent element, must equal the **parent menu name** of the submenu, where the submenu name is separated by a colon. If the menu has no parent, because its not a submenu, add a colon to the beginning of the name. The **type** attribute must be set to "MenuItem". Example for an UIElement defining a menu item:

```
 <UIElement Type="MenuItem">
  <!--the following element is a Local command ID-->
  <ID>3</ID>
  <Name>Open file from repository...</Name>
  <Place>-1</Place>
  <MenuID>129</MenuID>
  <Parent>:PlugIn Menu</Parent>
  <ImageID>0</ImageID>
 </UIElement>
```

XMLSpy 2005 makes it possible to add toolbar separators and menus if the value of the ID element is set to 0.

3.3 ATL sample files

The following pages show how to create a simple XMLSpy 2005 IDE plug-in DLL using ATL. To build the DLL it is necessary to know about ATL, the wizards that generate new ATL objects, as well as MS VisualStudio.

To access the API the implementation imports the Type Library of XMLSpy 2005. The code reads various properties and calls methods using the smart pointers provided by the #import statement.

In addition, the sample code uses the MFC class CString and the ATL conversion macros such as W2T.

At a glance the steps to create an ATL DLL are as follows:

1. Open VisualStudio and select "New..." from the "File" menu.
2. Select the "Projects" tab.
3. Select "ATL COM AppWizard" and type in a project name.
4. Select "Support for MFC" if you want to use MFC classes, or if you want to create a project for the sample code.

   Having created the project files you can add an ATL object to implement the IXMLSpyPlugIn interface:

1. Select "New ATL Object..." from the "Insert" menu.
2. Select "Simple Object" from the wizard and click "Next".
3. Type in a name for the object.
4. On the "Attributes" tab, select "Custom" for the type of interface, and disable Aggregation.

These steps produce the skeleton code for the implementation of the IDE plug-in interface. Please see the following pages on how to modify the code and achieve some basic functionality.

3.3.1 Interface description (IDL)

The IDL of the newly created ATL object contains a declaration for one COM interface.

- This interface declaration must be replaced by the declaration of IXMLSpyPlugIn as shown below.
- The IDL must also contain the definition of the SPYUpdateAction enumeration.
- Replace the generated default interface name, (created by the wizard) with "IXMLSpyPlugIn" in the coclass declaration. The IDL should then look something like the example code below:

Having created the ATL object, you then need to implement the IDE plug-in interface of XMLSpy 2005.

```
import "oaidl.idl";
import "ocidl.idl";

// ----- please insert the following block into your IDL file -----
 typedef enum {
  spyEnable = 1,
  spyDisable = 2,
  spyCheck = 4,
  spyUncheck = 8
 } SPYUpdateAction;

// ----- end insert block ----
```

 © 2005 Altova GmbH

```
// ----- E.g. Interface entry automatically generated by the ATL wizard -----
// [
//   object,
//   uuid(AB7CD86A-8145-429A-A1F3-270692EO8AFC),

//   helpstring("IXMLSpyPlugIn Interface")
//   pointer_default(unique)
// ]
// interface IXMLSpyPlugIn : IUnknown
// {
// };

// ----- end automatically generated Interface Entry


// ----- replace the Interface Entry (shown above) generated for you by the
ATL wizard, with the following block -----

 [
  odl,
  uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85),
  helpstring("IXMLSpyPlugIn Interface")
 ]
 interface IXMLSpyPlugIn : IUnknown
  {
   HRESULT _stdcall OnCommand([in] long nID, [in] IDispatch* pXMLSpy);

   HRESULT _stdcall OnUpdateCommand([in] long nID, [in] IDispatch* pXMLSpy,
[out, retval] SPYUpdateAction* pAction);

   HRESULT _stdcall OnEvent([in] long nEventID, [in] SAFEARRAY(VARIANT)*
arrayParameters, [in] IDispatch* pXMLSpy, [out, retval] VARIANT*
pReturnValue);

   HRESULT _stdcall GetUIModifications([out, retval] BSTR* pModificationsXML);

   HRESULT _stdcall GetDescription([out, retval] BSTR* pDescription);
  };

// ----- end replace block -----


// ----- The code below is automatically generated by the ATL wizard and will
look slightly different in your case -----

 [
 uuid(24FE0D1B-3FC0-494E-B36E-1D4CE412B014),
 version(1.0),
 helpstring("XMLSpyIDEPlugInDLL 1.0 Type Library")
 ]
 library XMLSPYIDEPLUGINDLLLib
 {
 importlib("stdole32.tlb");
 importlib("stdole2.tlb");

 [
  uuid(3800E791-7F6B-4ACD-9E32-2AC184444501),
  helpstring("XMLSpyIDEPlugIn Class")
 ]
 coclass XMLSpyIDEPlugIn
 {
  [default] interface IXMLSpyPlugIn;   // ----- define IXMLSpyPlugIn as the
default interface -----
 };
};
```

3.3.2 Class definition

In the class definition of the ATL object, several changes must be made. The class has to derive from IXMLSpyPlugIn, the "Interface Map" needs an entry for IXMLSpyPlugIn, and the methods of the IDE plug-in interface must be declared:

```
#ifndef __XMLSPYIDEPLUGIN_H_
#define __XMLSPYIDEPLUGIN_H_

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn
class ATL_NO_VTABLE CXMLSpyIDEPlugIn :
 public CComObjectRootEx<CComSingleThreadModel>,
 public CComCoClass<CXMLSpyIDEPlugIn, &CLSID_XMLSpyIDEPlugIn>,
 public IXMLSpyPlugIn
{
public:
 CXMLSpyIDEPlugIn()
 {
 }

DECLARE_REGISTRY_RESOURCEID(IDR_XMLSPYIDEPLUGIN)
DECLARE_NOT_AGGREGATABLE(CXMLSpyIDEPlugIn)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CXMLSpyIDEPlugIn)
 COM_INTERFACE_ENTRY(IXMLSpyPlugIn)
END_COM_MAP()

// IXMLSpyIDEPlugIn
public:
 virtual HRESULT _stdcall OnCommand(long nID, IDispatch* pXMLSpy);

 virtual HRESULT _stdcall OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction);

 virtual HRESULT _stdcall OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue);

 virtual HRESULT _stdcall GetUIModifications(BSTR* pModificationsXML);

 virtual HRESULT _stdcall GetDescription(BSTR* pDescription);
};

#endif //__XMLSPYIDEPLUGIN_H_
```

3.3.3 Implementation

The code below shows a simple implementation of an XMLSpy 2005 IDE plug-in. It adds a menu item and a separator (available with XMLSpy 2005) to the Tools menu. Inside the OnUpdateCommand() method, the new command is only enabled when the active document is displayed using the Grid View. The command searches for the XML element which has the current focus, and opens any URL starting with "http://", from the textual value of the element.

```
/////////////////////////////////////////////////////////////////////////////
// CXMLSpyIDEPlugIn

#import "XMLSpy.tlb"
using namespace XMLSpyLib;
```


© 2005 Altova GmbH

```
HRESULT CXMLSpyIDEPlugIn::OnCommand(long nID, IDispatch* pXMLSpy)
{
 USES_CONVERSION;

 if(nID == 1){
  IApplicationPtr ipSpyApp;

  if(pXMLSpy){
   if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void
**)&ipSpyApp))) {
    IDocumentPtr ipDocPtr = ipSpyApp->ActiveDocument;

    // we assume that grid view is active
    if(ipDocPtr){
     IGridViewPtr ipGridPtr = ipDocPtr->GridView;

     if(ipGridPtr) {
      IXMLDataPtr  ipXMLData = ipGridPtr->CurrentFocus;

      CString strValue = W2T(ipXMLData->TextValue);

      if(!strValue.IsEmpty() && (strValue.Left(7) == _T("http://")))
       ::ShellExecute(NULL,_T("open"),W2T(ipXMLData-
>TextValue),NULL,NULL,SW_SHOWNORMAL);
     }
    }
   }
  }
 }

 return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnUpdateCommand(long nID, IDispatch* pXMLSpy,
SPYUpdateAction* pAction)
{
 *pAction = spyDisable;

 if(nID == 1){
  IApplicationPtr ipSpyApp;

  if(pXMLSpy){
   if(SUCCEEDED(pXMLSpy->QueryInterface(__uuidof(IApplication),(void
**)&ipSpyApp))) {
    IDocumentPtr ipDocPtr = ipSpyApp->ActiveDocument;

    // only enable if grid view is active
    if((ipDocPtr != NULL) && (ipDocPtr->CurrentViewMode == spyViewGrid))
     *pAction = spyEnable;
   }
  }
 }

 return S_OK;
}


HRESULT CXMLSpyIDEPlugIn::OnEvent(long nEventID, SAFEARRAY **arrayParameters,
IDispatch* pXMLSpy, VARIANT* pReturnValue)
{
 return S_OK;
}

HRESULT CXMLSpyIDEPlugIn::GetUIModifications(BSTR* pModificationsXML)
```

```
{
 CComBSTR bstrMods = _T(" \
    <ConfigurationData> \
     <Modifications> ");
 // add "Open URL..." to Tools menu
 bstrMods.Append (_T(" \
      <Modification> \
       <Action>Add</Action> \
       <UIElement type=\"MenuItem\"> \
        <ID>1</ID> \
        <Name>Open URL...</Name> \
        <Place>0</Place> \
        <MenuID>129</MenuID> \
        <Parent>:Tools</Parent> \
       </UIElement> \
      </Modification> "));
 // add Seperator to Tools menu
 bstrMods.Append (_T(" \
      <Modification> \
       <Action>Add</Action> \
       <UIElement type=\"MenuItem\"> \
        <ID>0</ID> \
        <Place>1</Place> \
        <MenuID>129</MenuID> \
        <Parent>:Tools</Parent> \
       </UIElement> \
      </Modification> "));
 // finish modification description
 bstrMods.Append (_T(" \
     </Modifications> \
    </ConfigurationData>"));

 return bstrMods.CopyTo(pModificationsXML);
}


HRESULT CXMLSpyIDEPlugIn::GetDescription(BSTR* pDescription)
{
 CComBSTR bstrDescr = _T("ATL C++ XMLSpy IDE PlugIn;This PlugIn demonstrates
the implementation of a simple ATL DLL as a IDE PlugIn for XMLSpy.");
 return bstrDescr.CopyTo(pDescription);
}
```


© 2005 Altova GmbH

3.4 IXMLSpyPlugIn

See also

Methods
`OnCommand`
`OnUpdateCommand`
`OnEvent`
`GetUIModifications`
`GetDescription`

Description
If a DLL is added to XMLSpy 2005 as an IDE plug-in, it is necessary that it registers a COM component that answers to an IXMLSpyPlugIn interface with the reserved uuid(88F2A622-4B7E-42CD-8D04-3C0E5389DD85), for it to be recognized as a plug-in.

3.4.1 OnCommand

See also

Declaration: `OnCommand(nID as long, pXMLSpy as IDispatch)`

Description
The `OnCommand()` method of the interface implementation, is called each time a command added by the the IDE plug-in (menu item or toolbar button) is processed. nID stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy 2005.

Example

```
Public Sub IXMLSpyPlugIn_OnCommand(ByVal nID As Long, ByVal pXMLSpy As Object)
    If (Not (pXMLSpy Is Nothing)) Then
        Dim objDlg
        Dim objDoc As XMLSpyLib.Document
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "XML Files (*.xml)|*.xml|All Files (*.*)|*.*||"
            objDlg.FilterIndex = 1
            objDlg.ShowOpen

            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.Documents.OpenFile(objDlg.FileName, False)
                Set objDoc = Nothing
            End If
        End If

        If nID = 4 Or nID = 6 Then
            Set objDlg = CreateObject("MSComDlg.CommonDialog")
            objDlg.Filter = "All Files (*.*)|*.*||"
            objDlg.Flags = cdlOFNPathMustExist
            objDlg.ShowSave

            If Len(objDlg.FileName) > 0 Then
                Set objDoc = objSpy.ActiveDocument
```

```
                If Not (objDoc Is Nothing) Then
                    objDoc.SetPathName objDlg.FileName
                    objDoc.Save
                    Set objDoc = Nothing
                End If
            End If
        End If

        Set objSpy = Nothing
    End If
End Sub
```

3.4.2 OnUpdateCommand

See also

Declaration: `OnUpdateCommand(nID as long, pXMLSpy as IDispatch) as SPYUpdateAction`

Description
The `OnUpdateCommand()` method is called each time the visible state of a button or menu item needs to be set. `nID` stores the command ID defined by the `ID` element of the respective `UIElement`.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object.

Possible return values to set the update state are:

```
spyEnable         = 1
spyDisable        = 2
spyCheck          = 4
spyUncheck        = 8
```

Example

```
Public Function IXMLSpyPlugIn_OnUpdateCommand(ByVal nID As Long, ByVal pXMLSpy
As Object) As SPYUpdateAction
    IXMLSpyPlugIn_OnUpdateCommand = spyDisable

    If (Not (pXMLSpy Is Nothing)) Then
        Dim objSpy As XMLSpyLib.Application
        Set objSpy = pXMLSpy

        If nID = 3 Or nID = 5 Then
            IXMLSpyPlugIn_OnUpdateCommand = spyEnable
        End If
        If nID = 4 Or nID = 6 Then
            If objSpy.Documents.Count > 0 Then
                IXMLSpyPlugIn_OnUpdateCommand = spyEnable
            Else
                IXMLSpyPlugIn_OnUpdateCommand = spyDisable
            End If
        End If
    End If
End Function
```

3.4.3 OnEvent

See also

Declaration: `OnEvent(nEventID as long, arrayParameters as SAFEARRAY(VARIANT),`

© 2005 Altova GmbH

`pXMLSpy` as `IDispatch`) as VARIANT

Description
`OnEvent()` is called each time an event is raised from XMLSpy 2005.

Possible values for `nEventID` are:

On_BeforeStartEditing	= 1
On_EditingFinished	= 2
On_FocusChanged	= 3
On_Beforedrag	= 4
On_BeforeDrop	= 5
On_OpenProject	= 6
On_OpenDocument	= 7
On_CloseDocument	= 8
On_SaveDocument	= 9

Events available since XMLSpy 2005 4r4:

On_DocEditDragOver	= 10
On_DocEditDrop	= 11
On_DocEditKeyDown	= 12
On_DocEditKeyUp	= 13
On_DocEditKeyPressed	= 14
On_DocEditMouseMove	= 15
On_DocEditButtonUp	= 16
On_DocEditButtonDown	= 17
On_DocEditContextMenu	= 18
On_DocEditPaste	= 19
On_DocEditCut	= 20
On_DocEditCopy	= 21
On_DocEditClear	= 22
On_DocEditSelectionChanged	= 23

Events available since XMLSpy 2005 2004:

On_DocEditDragOver	= 10

Events available since XMLSpy 2005 2004r4 (type library version 1.4):

On_BeforeOpen Project	= 25
On_BeforeOpenDocument	= 26
On_BeforeSaveDocument	= 27
On_BeforeCloseDocument	= 28
On_ViewActivation	= 29
On_DocEditKeyboardEvent	= 30
On_DocEditMouseEvent	= 31

The names of the events are the same as they appear in the Scripting Environment of XMLSpy 2005. For IDE plug-ins the names used are immaterial. The events are identified using the `ID` value.

`arrayParameters` is an array which is filled with the parameters of the currently raised event. Order, type and meaning of the single parameters are available through the scripting environment of XMLSpy 2005. The events module of a scripting project, contains predefined functions for all events prior to version 4.4. The parameters passed to the predefined functions are identical to the array elements of the `arrayParameters` parameter.

Events raised from the Authentic of XMLSpy 2005 do not pass any parameters directly. An "event" object is used instead. The event object can be accessed through the Document object of the active document.

`pXMLSpy` holds a reference to the dispatch interface of the `Application` object of XMLSpy 2005.

If the return value of `OnEvent()` is set, then neither the IDE plug-in, nor an event handler inside of the scripting environment will get this event afterwards. Please note that all IDE plug-ins get/process the event before the Scripting Environment does.

3.4.4 GetUIModifications

See also

Declaration: `GetUIModifications()` as String

Description
The `GetUIModifications()` method is called during initialization of the plug-in, to get the configuration XML data that defines the changes to the UI of XMLSpy 2005. The method is called when the plug-in is loaded for the first time, and at every start of XMLSpy 2005.

See also Configuration XML for a detailed description how to change the UI.

Example

```
Public Function IXMLSpyPlugIn_GetUIModifications() As String
    ' GetUIModifications() gets the XML file with the specified modifications
of
    ' the UI from the config.xml file in the plug-in folder
    Dim strPath As String
    strPath = App.Path

    If Len(strPath) > 0 Then
        Dim fso As New FileSystemObject
        Dim file As file
```

 © 2005 Altova GmbH

```
        Set file = fso.GetFile(strPath & "\config.xml")

        If (Not (file Is Nothing)) Then
            Dim stream As TextStream
            Set stream = file.OpenAsTextStream(ForReading)

            ' this replaces the token '**path**' from the XML file with
            ' the actual installation path of the plug-in to get the image
file
            Dim strMods As String
            strMods = stream.ReadAll
            strMods = Replace(strMods, "**path**", strPath)

            IXMLSpyPlugIn_GetUIModifications = strMods
        Else
            IXMLSpyPlugIn_GetUIModifications = ""
        End If
    End If
End Function
```

3.4.5 GetDescription

See also

Declaration: `GetDescription()` as String

Description

`GetDescription()` is used to define the description string for the plug-in entries visible in the Customize dialog box.

Example

```
Public Function IXMLSpyPlugIn_GetDescription() As String
    IXMLSpyPlugIn_GetDescription = "Sample Plug-in for XMLSpy;This Plug-in
demonstrates the implementation of a simple VisualBasic DLL as a Plug-in for
XMLSpy."
End Function
```

4 The XMLSpy API

The COM-based XMLSpy API of XMLSpy 2005 enables other applications to use the functionality of XMLSpy 2005. As a result, it is now possible to automate a wide range of tasks, from validating an XML file to modifying complex XML content (with the XMLData interface). The XMLSpy API is also used by the built-in Scripting Environment of XMLSpy 2005 to access application functionality (for a complete description of which, see Scripting). It is also used by and by IDE Plugins.

XMLSpy 2005 and the XMLSpy API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the XMLSpy API from common development environments, such as those using C, C++, VisualBasic, and Delphi, and with scripting languages like JavaScript and VBScript.

The following limitations must be considered in your client code:

- Be aware that if your client code crashes, instances of XMLSpy 2005 may still remain in the system.
- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Terminate XMLSpy 2005 with the Application.Quit method.
- Don't forget to disable dialogs if the user interface is not visible.
- See Error handling for details of how to avoid annoying error messages.
- Free references explicitly if you are using C or C++.

This documentation
The documentation about the XMLSpy API is broadly divided into two parts.

- The first part consists of an Overview, which describes the object model for the API, important concepts, and Usage Examples to help you get started.
- The second part is a reference section that contains descriptions of all the interface objects of the XMLSpy API.

There have been improvements and updates made from release to release. The differences between releases are documented in Release Notes.

© 2005 Altova GmbH

4.1 Overview

This overview of the XMLSpy API provides you with the object model for the API, and with a description of the most important API processes. The following topics are covered:

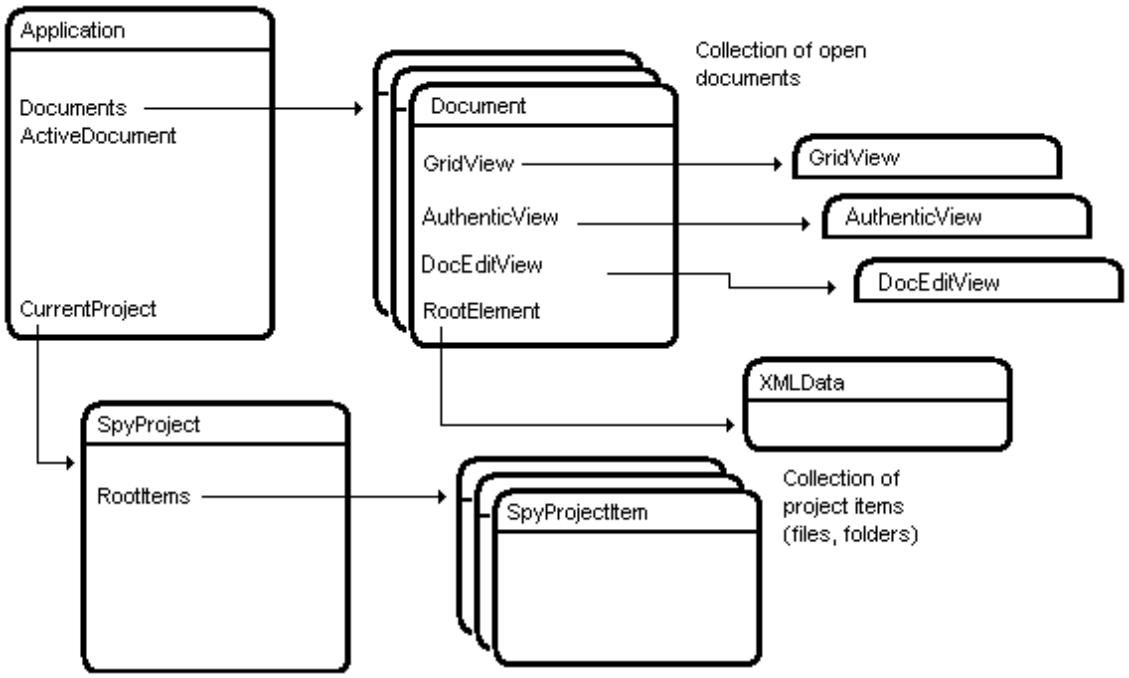
- The object model
- Simple document access
- Error handling
- Events
- Import and export of data
- Using XMLData to modify document structure
- The DOM and XMLData

4.1.1 Object model

The starting point for every application which uses the XMLSpy 2005 API is the `Application` object. This object contains general methods like import/export support and references to the open documents and any open project.

To create an instance of the `Application` object, call `CreateObject("XMLSpy.Application")` from VisualBasic or a similar function from your preferred development environment to create a COM object. There is no need to create any other objects, to use the complete XMLSpy API (It is in fact not even possible). All other interfaces are accessed through other objects with the Application object as the starting point.

The picture below shows you the links between the main objects of the XMLSpy 2005 API:



The application object consists of the following parts:

1. Document collection and reference to the active document.
2. Reference to current project and methods for creating and opening of projects.

3. Methods to support the export to and import from databases, text files and Word documents.

4. URL management.

5. Methods for macro menu items.

Once you have created an Application object you can start using the functionality of XMLSpy 2005. Most of the times you either open a project and access the documents from there or you directly open a document via the `Documents` interface.

4.1.2 Simple document access

Create and open files
Use the methods of the `Documents` interface to create and open documents. This interface is accessible via the `Application.Documents` property.

Example:

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile("C:\xmlfiles\myfile.xml", False)
```

Sometimes it is necessary to show a file dialog to the user to select a file. This usually happens in the scripting environment and you only need to pass True as the second parameter to `OpenFile()`.

To create a new file use `Documents.NewFile()`. Any existing file with the same path will be overwritten during the next `Document.Save()` call.

Documents collection
All open documents are accessible via the `Documents` collection. All collection objects in the XMLSpy 2005 API conform to the Microsoft specifications. So it is possible to use the For-Each loop in VisualBasic to iterate through the open documents.

Example:

```
Dim objDocs As Documents
Dim objDoc As Document

Set objDocs = objSpy.Documents

For Each objDoc In objDocs
 'do something useful with your document
 objDoc.AssignXSL "C:\myXSL.xsl", False
Next
```

Another way to access a document is the `Documents.Item` method. The method takes an index as a parameter and gets the corresponding document object from the collection. Please note that 1 is the first index value. `Documents.Count` retrieves the total number of documents.

Example:

```
Dim objDocs As Documents
Dim objDoc As Document

Set objDoc = objDocs.Item(1) 'gets the first document
```

Validation

 © 2005 Altova GmbH

One common task on documents is to validate them against an assigned schema or DTD. If the XML file has no schema or DTD already assigned, use `Document.AssignSchema` or `Document.AssignDTD` to add the necessary references to the document.

Examples:

```
objSpy.ActiveDocument.AssignSchema "C:\mySchema.xsd", False

 Or

objSpy.ActiveDocument.AssignDTD "C:\myDTD.dtd", False
```

If you want the user to select a schema or DTD, pass True as the second parameter to these functions to display a file-dialog. These methods only put the reference into the document and do not check the existence of the specified file. If the file path is not valid, the validation will fail.

After you have assigned a valid schema or DTD reference to your file, you are able to validate it with `Document.IsValid`. IsValid needs some out-parameters that must be declared as VARIANTs to be accessible from script languages like VBScript and JavaScript.

Example:

```
Dim bValid   As Boolean
Dim strMsg   As Variant
Dim nPos  As Variant
Dim objBadXMLData As Variant

bValid = objSpy.ActiveDocument.IsValid(strMsg, nPos, objBadXMLData)

If bValid = False Then
 a = MsgBox("The document is not valid:" & Chr(13) &
   strMsg & Chr(13) & "position: " & nPos &
   Chr(13) & "XMLData name: " & objBadXMLData.Name)
Else
 a = MsgBox("The document is valid")
End If
```

4.1.3 Error handling

The XMLSpy 2005 API returns errors in two different ways. Every API method returns an `HRESULT`. This return value informs the caller about any malfunctions during the execution of the method. If the call was successful, the return value is equal to `S_OK`. C/C++ programmers generally use `HRESULT` to detect any errors.

VisualBasic, scripting languages and other high-level development environments, don't give the programmer access to the returning `HRESULT` of a COM call. They use the second error raising mechanism also supported by the XMLSpy 2005 API, the `IErrorInfo` interface. If an error occurs the API creates a new object that implements the `IErrorInfo` interface. The development environment takes this interface, and fills its own error handling mechanism with the provided information.

The next paragraphs describes how to deal with errors raised from the XMLSpy 2005 API in different development environments.

VisualBasic
A common way to handle errors in VisualBasic is to define an error handler. This error handler can be set with the `On Error` statement. Usually the handler displays a error message and does some cleanup to avoid spare references and any kind of resource leaks. VisualBasic fills its own `Err` object with the information from the `IErrorInfo` interface.

Example:

```
Sub Validate()
 'place variable declarations here

 'set error handler
 On Error GoTo ErrorHandler

 'if IsValid() fails, program execution continues
            'at ErrorHandler:
 bValid = objSpy.ActiveDocument.IsValid(strMsg,nPos,objBadXMLData)

 'additional code comes here

 'exit
 Exit Sub

 ErrorHandler:
 Set objBadXMLData = Nothing

 MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
  "Description: " & Err.Description)
End Sub
```

JavaScript
The Microsoft implementation of JavaScript (JScript) provides a try-catch mechanism to deal with errors raised from COM calls. It is very similar to the VisualBasic approach, because you also declare an error object containing the necessary information.

Example:

```
Function Validate()
{
 // please insert variable declarations here

 try{
  bValid = objSpy.ActiveDocument.IsValid(sMsg,nPos,objBad);
 }
 catch(Error) {
  objBad = Null;
  sError = Error.description;
  nErrorCode = Error.number;
  Return False;
 }

 Return bValid;
}
```

C/C++
C/C++ gives you easy access to the HRESULT of the COM call and to the `IErrorInterface`.

```
HRESULT hr;

// Call IsValid() from the XMLSpy API
If(FAILED(hr = ipDoc->IsValid(&varMsg,&varPos,&varBadData)))      {
 IErrorInfo *ipErrorInfo = Null;

 If(SUCCEEDED(::GetErrorInfo(0,&ipErrorInfo)))      {
  BSTRbstrDescr;
  ipErrorInfo->GetDescription(&bstrDescr);

  // handle Error information
  wprintf(L"Error message:\t%s\n",bstrDescr);
  ::SysFreeString(bstrDescr);
```

 © 2005 Altova GmbH

```
   // release Error info
   ipErrorInfo->Release();
  }
 }
```

4.1.4 Events

The automation interface of XMLSPY provides a large set of events to allow for a tight integration of XMLSPY and its clients. The way events can be used by clients depends on their technology used to connect to XMLSPY and on the clients programming language.

XMLSPY scripting environment - macros, forms and event handlers written with the FormEditor

The XMLSPY scripting environment automatically invokes the event handler with the corresponding name. Use the FormEditor to add code for those events you want to handle. The FormEditor predefines the correct signatures for all event handlers when you create a new project. For more information see the scripting environment.

VBScript

All event handler functions start with 'On_'. To stay compatible to existing events, their name might differ slightly from the name used on the connection point interface. Specification of event parameters is optional so that old event handlers that did not expect any parameters still work.

JScript

All event handler functions start with 'On_'. To stay compatible to existing events, their name might differ slightly from the name used on the connection point interface. Specification of event parameters is optional so that old event handlers that did not expect any parameters still work.

XMLSPY IDE Plugin

XMLSPY Plugins get informed about outstanding events by a call to the method IXMLSpyPlugIn.OnEvent. Different events are identified by a unique number. Parameters are passed to the event handler packed into a save-array.
Alternatively, Plugins can use the connection point mechanism to request events on specific objects like any other external client. See below for more information.

External clients

COM specifies the connection point mechanism to define a way how a client can register itself at a server for callbacks. The automation interface for XMLSPY defines the necessary event interfaces. The way to connect to those events, depends on the programming language you use in your client.

VBA - VisualBasic for Applications

Use the Dim WithEvents statement to receive events from a dedicated object instance.

```
' tell VBA to connect to events once this objects gets created
Dim WithEvents objView As XMLSpyLib.AuthenticView
' initialize the object somewhere in your code
...
Set objView = objSpy.ActiveDocument.AuthenticView
...
' sample event handler for the OnMouseEvent of the object in objView
Private Function objView_OnMouseEvent (ByVal i_nXPos As Long, ByVal
i_nYPos As Long,
                                       ByVal i_eMouseEvent As
XMLSpyLib.SPYMouseEvent,
                                       ByVal i_pRange As
XMLSpyLib.IAuthenticRange ) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
```

```
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    End If
End Function
```

Windows Scripting Technologies - VBScript

Use the method WScript.ConnectObject to receive events.

```
' the event handler function
Function DocEvent_OnBeforeCloseDocument(objDocument)
   Call WScript.Echo("received event - before closing document")
End Function

' create or connect to XMLSPY
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objFSO = WScript.CreateObject("Scripting.FileSystemObject")
Set objSpy = WScript.GetObject("", "XMLSpy.Application")

' create document object and connect to its events
objSpy.Visible = True
Set objDoc = objSpy.Documents.OpenFile ("C:\\Program
Files\\Altova\\XMLSPY2004\\Examples\\OrgChart.xml", False)
Call WScript.ConnectObject(objDoc, "DocEvent_")

' keep running while waiting on the event
' in the meantime close the document in XMLSPY manually
Call WScript.Echo ("sleeping for 10 seconds ...")
Call WScript.Sleep (10000)

Set objDoc = Nothing
Call WScript.Echo ("stopped listening for event")
Call objSpy.Quit
```

Windows Scripting Technologies - JScript

Use the method WScript.ConnectObject to receive events.

```
// the event handler function
function DocEvent_OnBeforeCloseDocument(objDocument)
{
   WScript.Echo("received event - before closing document");
}

// create or connect to XMLSPY
try
{
   // create the environment and XMLSPY
   objWshShell = WScript.CreateObject("WScript.Shell");
   objFSO = WScript.CreateObject("Scripting.FileSystemObject");
   objSpy = WScript.GetObject("", "XMLSpy.Application");
}
catch(err)
   { WScript.Echo ("Can't create WScript.Shell object or XMLSPY"); }

// create document object and connect to its events
objSpy.Visible = true;
objDoc = objSpy.Documents.OpenFile ("C:\\Program
Files\\Altova\\XMLSPY2004\\Examples\\OrgChart.xml", false);
WScript.ConnectObject(objDoc, "DocEvent_");

// keep running while waiting on the event
```


© 2005 Altova GmbH

```
// in the meantime close this document in XMLSPY manually
WScript.Echo ("sleeping for 10 seconds ...");
WScript.Sleep (10000);

objDoc = null;
WScript.Echo ("stopped listening for event");
objSpy.Quit();
```

C++

The following is an excerpt of a C++ client using connection points via ATL

```
#import "XMLSpy.tlb"
using namespace XMLSpyLib;

static const int XMLSPYSE_APPLICATION_SOURCE_ID = 0;
class XMLSpySEApplicationEventPump;

// nested class to receive application events
typedef IDispEventImpl< XMLSPYSE_APPLICATION_SOURCE_ID,
                        XMLSpySEApplicationEventPump,
                        &DIID__IApplicationEvents,
                        &LIBID_XMLSpyLib, TYPELIB_MAJ, TYPELIB_MIN >
          XMLSpySEApplicationEventImpl;

// the class receiving the application events using a sink map
class XMLSpySEApplicationEventPump :
   public XMLSpySEApplicationEventImpl
{
   private:
          static _ATL_FUNC_INFO Info_OnBeforeOpenProject;

   public:
          XMLSpySEApplicationEventPump ();
          ~XMLSpySEApplicationEventPump();

   // connection point events on XMLSpySEApplicationEventImpl
   public:
          void __stdcall OnBeforeOpenProject (IFileSelectionDlg *io_ipDlg);

   BEGIN_SINK_MAP(XMLSpySEApplicationEventPump)
          // OnBeforeOpenProject
          SINK_ENTRY_INFO(XMLSPYSE_APPLICATION_SOURCE_ID,
DIID__IApplicationEvents,
                                  1, OnBeforeOpenProject,
&Info_OnBeforeOpenProject)
   END_SINK_MAP()
};

// we use here explicit declaration of event signature to avoid loading
the typelib
_ATL_FUNC_INFO XMLSpySEApplicationEventPump::Info_OnBeforeOpenProject =
   {CC_STDCALL, VT_EMPTY, 1, {VT_DISPATCH}};

XMLSpySEApplicationEventPump::XMLSpySEApplicationEventPump ()
{
   ...
   // get an IUnknown pointer to XMLSpy application from somewhere and
store it in ipXMLSPY
   HRESULT hRes = XMLSpySEApplicationEventImpl::DispEventAdvise(ipXMLSPY,
&DIID__IApplicationEvents);
   ...
}

XMLSpySEApplicationEventPump::~XMLSpySEApplicationEventPump()
{
```

```
   ...
   // disconnect from connection point
   HRESULT hRes =
XMLSpySEApplicationEventImpl::DispEventUnadvise(ipXMLSPY,
&DIID__IApplicationEvents);
   ...
}

// the handler for the OnBeforeOpenProject event
void __stdcall XMLSpySEApplicationEventPump::OnBeforeOpenProject
(IFileSelectionDlg *io_ipDlg)
{
   // do something with io_ipDlg here
}
```

4.1.5 Import and export of data

Before you implement your import and export tasks with the XMLSpy 2005 API, it is a good practice to test the connections, parameters, SQL queries and so on in XMLSpy 2005. This way, you are able to verify the results and to make quick adjustments to all import or export parameters.

Most of the methods for importing and exporting data are placed in the `Application` object, the remaining functions are accessible via the `Document` interface.

There is some preparatory work necessary, before the actual import or export can be started. Every import/export job consists of two parts. You need to define a connection to your data and the specific behaviour for the import/export process.

In case of an import, the connection is either a database, a text-file or a Word document. The behaviour is basically which data (columns) should be imported in XMLSpy 2005.

In case of an export, the connection is either a database or a text file. Specify which data (elements of the XML file) and additional parameters (e.g. automatic key generation or number of sub-levels) to use from the XML-structure for the behaviour.

The properties in the `DatabaseConnection`, `TextImportExportSettings` and `ExportSettings` interfaces have default values. See the corresponding descriptions in the `Interfaces` chapter for further information.

Import from database
These are the steps to establish a connection to an existing database for import:

1. Use a `DatabaseConnection` object and set the properties:
   The method `Application.GetDatabaseSettings` returns a new object for a database connection:

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
```

You have to set either an **ADO connection string**,

- `objImpSettings.ADOConnection = strADOConnection`

or the **path** to an existing database file:

- `objImpSettings.File = "C:\myDatabase.mdb"`

To complete the settings you create a **SQL select statement** to define the data to be queried:

 © 2005 Altova GmbH

- `objImpSettings.SQLSelect = "SELECT * FROM myTable"`

2. Call `Application.GetDatabaseImportElementList` to get a collection of the resulting columns of the SQL query:

```
Dim objElementList As ElementList
Set objElementList =
objSpy.GetDatabaseImportElementList(objImpSettings)
```

This collection gives you the opportunity to control which columns should be imported and what type the new elements will become. Each item of the collection represents one column to import. If you remove an item, the corresponding column will not be imported. You can additionally modify the `ElementListItem.ElementKind` property, to set the type of the created XML elements for each column.

Please consider that `GetDatabaseImportElementList()` executes the SQL query and could initiate a time consuming call. To avoid this, it is possible to pass a null-pointer (Nothing in VisualBasic) as the second parameter to `ImportFromDatabase()` to import all columns as plain XML elements.

3. Start the import with `Application.ImportFromDatabase`:

```
Dim objImpDoc As Document
Set objImpDoc =
objSpy.ImportFromDatabase(objImpSettings,objElementList)
```

Import from Text
Importing data from a text file is similar to the import from a database. You must use other interfaces (described in steps 1-3 below) with different methods and properties:

1. Use a `TextImportExportSettings` object and set the properties:
The method `Application.GetTextImportExportSettings` returns a new object to specify a text file for import.

```
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings
```

You have to set at least the `ImportFile` property to the path of the file for the import. Another important property is `HeaderRow`. Set it to False, if the text file does not contain a leading line as a header row.

```
objImpSettings.ImportFile = "C:\myFile.txt"
objImpSettings.HeaderRow = False
```

Call `Application.GetTextImportElementList` to get a collection of all columns inside the text file:

```
Dim objElementList As ElementList
Set objElementList = objSpy.GetTextImportElementList(objImpSettings)
```

3. Start the import with `Application.ImportFromText`:

```
Dim objImpDoc As Document
Set objImpDoc = objSpy.ImportFromText(objImpSettings,objElementList)
```

Export to database

1. Use a `DatabaseConnection` object and set the necessary properties.
   All properties except `SQLSelect` are important for the export. `ADOConnection` or `File` defines the target for the output. You need to set only one of them.

2. Fill an `ExportSettings` object with the required values.
   These properties are the same options as those available in the export dialog of XMLSpy 2005. Select the menu option **Convert | Export to Text files/Database** to see the options and try a combination of export settings. After that it is easy to transfer these settings to the properties of the interface.

   Call "Application.GetExportSettings" to get a ExportSettings object:

   ```
   Dim objExpSettings As ExportSettings
   Set objExpSettings = objSpy.GetExportSettings


   objExpSettings.CreateKeys = False
   objExpSettings.ExportAllElements = False
   objExpSettings.SubLevelLimit = 2
   ```

3. Build an element list with `Document.GetExportElementList`.
   The element list enables you to eliminate XML elements from the export process. It also gives you information about the record and field count in the `RecordCount` and `FieldCount` properties. Set the `ExportSettings.ElementList` property to this collection. It is possible to set the element list to null/Nothing (default) to export all elements.

4. Call `Document.ExportToDatabase` to execute the export.
   The description of the `ExportToDatabase` method contains also a code example for a database export.

Export to text

1. Use a `TextImportExportSettings` object and set the necessary properties.

2. Fill an `ExportSettings` object with the required values.
   See item number 2 from "Export to database" earlier on this page.

3. Build an element list with `Document.GetExportElementList`.
   See item number 3 from "Export to database" earlier on this page.

4. Call `Document.ExportToText` to execute the export.
   The description of the `ExportToText` method contains also a code example for a database export.

© 2005 Altova GmbH

4.1.6 Using XMLData to modify document structure

XMLData gives you access to the elements of an currently open XML file. It enables you to perform all necessary modifications to the elements of the XML structure. The main functionality of `XMLData` is:

1. Access to the names and values of all kinds of elements (e.g. elements, attributes)
2. Creation of new elements of all kinds.
3. Insertion and appending of new elements.
4. Erasing of existing child elements.

Structure of XMLData
Before you can use the `XMLData` interface, you have to know how an existing XML file is mapped into a `XMLData` structure. One major thing you must be aware of is, that `XMLData` has no separate branch of objects for attributes.

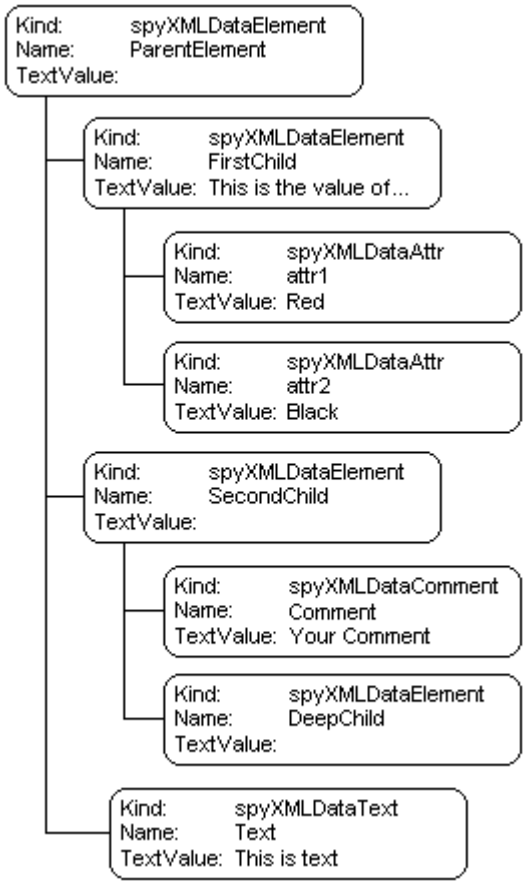
The attributes of an element are also children of the element. The `XMLData.Kind` property, gives you the opportunity to distinguish between the different types of children of an element.

Example:

This XML code,

```
<ParentElement>
 <FirstChild attr1="Red" attr2="Black">
  This is the value of FirstChild
 </FirstChild>
 <SecondChild>
  <!--Your Comment-->
  </DeepChild>
 </SecondChild>
 This is Text
</ParentElement>
```

is mapped to the following XMLData object structure:



The parent of all XML elements inside of a document is the property `Document.RootElement`. Use this `XMLData` object to get references to all other XML elements in the structure.

Name and value of elements
To get and to modify the name and value of all types of XML elements use the `XMLData.Name` and `XMLData.TextValue` properties. It is possible that several kinds of `XMLData` objects and empty elements do not have a text value associated.

Creation and insertion of new XMLData objects
The creation of a new XML language entity requires the following steps:

1. Create the new `XMLData` object:
   Use the `Document.CreateChild` method to create a new `XMLData` object. Set name and value after you have inserted the new XML entity (see point 3).
2. Find the correct location for the new XMLData object:
   To insert a new `XMLData` object you have to get a reference to the parent first. If the new child is to become the last child of the parent, use the `XMLData.AppendChild` method to insert the `XMLData` object. If the new child should be located elsewhere in the sequence of child objects, use the `XMLData.GetFirstChild` and `XMLData.GetNextChild` to move the iterator to the child before which the new child should be inserted.
3. Insert the new child with `XMLData.InsertChild.` The new child will be inserted immediately before the current child.

The following example adds a third child between `<FirstChild>` and the `<SecondChild>`

 © 2005 Altova GmbH

element:

```
Dim objParent As XMLData
Dim objChild As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)

'objParent is set to <ParentElement>
'GetFirstChild(-1) gets all children of the parent element
'and move to <SecondChild>
Set objChild = objParent.GetFirstChild(-1)
Set objChild = objParent.GetNextChild

objParent.InsertChild objNewChild
objNewChild.Name = "OneAndAHalf"
Set objNewChild = Nothing
```

Child elements should be inserted in a special order. Please avoid inserting attributes into a sequence after any other child elements, i.e. make sure that attributes are placed first.

Copying of existing XMLData objects
If you want to insert existing `XMLData` objects at a different place in the same document or into another document you can't use the `XMLData.InsertChild` and `XMLData.AppendChild` methods. These methods only work for new `XMLData` objects.

Instead of using `InsertChild` or `AppendChild` you have to copy the object hierarchy manually. The following function written in JavaScript is an example for recursively copying `XMLData`:

```
// this Function returns a complete copy of the XMLData Object
Function GetCopy(objXMLData)
{
 var objNew;
 objNew = Application.ActiveDocument.CreateChild(objXMLData.Kind);

 objNew.Name = objXMLData.Name;
 objNew.TextValue = objXMLData.TextValue;

 If(objXMLData.HasChildren){
  var objChild;
  objChild = objXMLData.GetFirstChild(-1);

  While(objChild) {
   try {
    objNew.AppendChild(GetCopy(objChild));
    objChild = objXMLData.GetNextChild();
   }
   catch(e) {
    objChild = Null;
   }
  }
 }

 Return objNew;
}
```

Erasing of XMLData objects
`XMLData` provides two methods for the deletion of child objects, `XMLData.EraseAllChildren` and `XMLData.EraseCurrentChild`.

To erase `XMLData` objects you need access to the parent of the elements you want to remove.

Use `XMLData.GetFirstChild` and `XMLData.GetNextChild` to get a reference to the parent `XMLData` object.

See the method descriptions of `EraseAllChildren` and `EraseCurrentChild` for examples how to erase XML elements.

4.1.7 The DOM and XMLData

The `XMLData` interface gives you full access to the XML structure behind the current document with less methods than DOM and is much simpler. The `XMLData` interface is a minimalist approach to reading and modifying existing, or newly created XML data. You might however, want to use a DOM tree because you can access one from an external source or you just prefer the MSXML DOM implementation.

The `ProcessDOMNode()` and `ProcessXMLDataNode()` functions provided below convert any segments of an XML structure between `XMLData` and DOM.

To use the `ProcessDOMNode()` function:

- pass the root element of the DOM segment you want to convert in `objNode` and
- pass the plugin object with the CreateChild() method in `objCreator`

To use the `ProcessXMLDataNode()` function:

- pass the root element of the `XMLData` segment in `objXMLData` and
- pass the `DOMDocument` object created with MSXML in `xmlDoc`

```
////////////////////////////////////////////////////////////
// DOM To XMLData conversion
Function ProcessDOMNode(objNode,objCreator)
{
 var objRoot;
 objRoot = CreateXMLDataFromDOMNode(objNode,objCreator);

 If(objRoot) {
  If((objNode.nodeValue != Null) && (objNode.nodeValue.length > 0))
   objRoot.TextValue = objNode.nodeValue;
   // add attributes
  If(objNode.attributes) {
   var Attribute;
   var oNodeList = objNode.attributes;

   For(var i = 0;i < oNodeList.length; i++) {
    Attribute = oNodeList.item(i);

    var newNode;
    newNode = ProcessDOMNode(Attribute,objCreator);

    objRoot.AppendChild(newNode);
   }
  }
  If(objNode.hasChildNodes){
   try {
    // add children
    var Item;
    oNodeList = objNode.childNodes;

    For(var i = 0;i < oNodeList.length; i++) {
       Item = oNodeList.item(i);
```

 © 2005 Altova GmbH

```
      var newNode;
      newNode = ProcessDOMNode(Item,objCreator);

      objRoot.AppendChild(newNode);
     }
    }
    catch(err) {
    }
   }
  }
  Return objRoot;
}

Function CreateXMLDataFromDOMNode(objNode,objCreator)
{
 var bSetName = True;
 var bSetValue = True;

 var nKind = 4;

 switch(objNode.nodeType){
  Case 2:nKind = 5;break;
  Case 3:nKind = 6;bSetName = False;break;
  Case 4:nKind = 7;bSetName = False;break;
  Case 8:nKind = 8;bSetName = False;break;
  Case 7:nKind = 9;break;
 }
 var objNew = Null;
 objNew = objCreator.CreateChild(nKind);

 If(bSetName)
  objNew.Name = objNode.nodeName;

 If(bSetValue && (objNode.nodeValue != Null))
  objNew.TextValue = objNode.nodeValue;

 Return objNew;
}
////////////////////////////////////////////////////////////
// XMLData To DOM conversion

Function ProcessXMLDataNode(objXMLData,xmlDoc)
{
 var objRoot;
 objRoot = CreateDOMNodeFromXMLData(objXMLData,xmlDoc);

 If(objRoot) {
  If(IsTextNodeEnabled(objRoot) && (objXMLData.TextValue.length > 0))
   objRoot.appendChild(xmlDoc.createTextNode(objXMLData.TextValue));

  If(objXMLData.HasChildren){
   try {
    var objChild;
    objChild = objXMLData.GetFirstChild(-1);

    While(True) {
     If(objChild) {
      var newNode;
      newNode = ProcessXMLDataNode(objChild,xmlDoc);

      If(newNode.nodeType == 2){
       // child node is an attribute
       objRoot.attributes.setNamedItem(newNode);
      }
      Else
       objRoot.appendChild(newNode);
```

```
      }
      objChild = objXMLData.GetNextChild();
     }
    }
    catch(err) {
    }
   }
  }
  Return objRoot;
}

Function CreateDOMNodeFromXMLData(objXMLData,xmlDoc)
{
 switch(objXMLData.Kind){
  Case 4:Return xmlDoc.createElement(objXMLData.Name);
  Case 5:Return xmlDoc.createAttribute(objXMLData.Name);
  Case 6:Return xmlDoc.createTextNode(objXMLData.TextValue);
  Case 7:Return xmlDoc.createCDATASection(objXMLData.TextValue);
  Case 8:Return xmlDoc.createComment(objXMLData.TextValue);
  Case 9:Return
xmlDoc.createProcessingInstruction(objXMLData.Name,objXMLData.TextValue);
 }

 Return xmlDoc.createElement(objXMLData.Name);
}
Function IsTextNodeEnabled(objNode)
{
 switch(objNode.nodeType) {
  Case 1:
  Case 2:
  Case 5:
  Case 6:
  Case 11:Return True;
 }
 Return False;
}
```

4.1.8 Obsolete Authentic Row operations

If the schema on which an XML document is based specifies that an element is repeatable, such a structure can be represented in Authentic as a table. When represented as a table, rows and their contents can be manipulated individually, thereby allowing you to manipulate each of the repeatable elements individually. Such row operations would be performed by an external script.

If an external script is to perform row operations then two steps must occur:

- The first step checks whether the cursor is currently in a row using a property. Such a check could be, for example, `IsRowInsertEnabled`, which returns a value of either TRUE or FALSE.
- If the return value is TRUE then a row method, such as `RowAppend`, can be called. (`RowAppend` has no parameters and returns no value.)

The following is a list of properties and methods available for table operations. Each property returns a BOOL, and the methods have no parameter.

 © 2005 Altova GmbH

Property	Method	Table operations
`IsRowInsertEnabled`	**RowInsert**, superseded by `AuthenticRange.InsertRow`	Insert row operation
`IsRowAppendEnabled`	**RowAppend**, superseded by `AuthenticRange.AppendRow`	Append row operation
`IsRowDeleteEnabled`	**RowDelete**, superseded by `AuthenticRange.DeleteRow`	Delete row operation
`IsRowMoveUpEnabled`	**RowMoveUp**, superseded by `AuthenticRange.MoveRowUp`	Move XML data up one row
`IsRowMoveDownEnabled`	**RowMoveDown**, superseded by `AuthenticRange.MoveRowDown`	Move XML data down one row
`IsRowDuplicateEnabled`	**RowDuplicate**, superseded by `AuthenticRange.DuplicateRow`	Duplicate currently selected row

4.1.9 Obsolete Authentic Editing operations

When XML data is displayed as data in Authentic, it is possible to manipulate individual elements using standard editing operations such as cut, copy, and paste. However, not all XML data nodes can be edited. So, in order to carry out an editing operation, first a property is used to test whether editing is possible, and then a method is called to perform the editing operation.

The only method that does not have a test is the method `EditSelectAll`, which automatically selects all elements displayed in the document.

The following is a list of properties and methods that perform editing operations. Each property returns a BOOL, and the methods have no parameter.

Property	Method	Editing operation
`IsEditUndoEnabled`	**EditUndo**, superseded by `AuthenticView.Undo`	Undo an editing operation
`IsEditRedoEnabled`	**EditRedo**, superseded by `AuthenticView.Redo`	Redo an editing operation
`IsEditCopyEnabled`	**EditCopy**, superseded by `AuthenticRange.Copy`	Copy selected text to the clipboard
`IsEditCutEnabled`	**EditCut**, superseded by `AuthenticRange.Cut`	Cut selected text to the clipboard
`IsEditPasteEnabled`	**EditPaste**, superseded by `AuthenticRange.Paste`	Paste from clipboard to current cursor position
`IsEditClearEnabled`	**EditClear**, superseded by `AuthenticRange.Delete`	Clear selected text from XML document

4.2 Usage Examples

The automation interface of XMLSpy 2005 can be used in many different ways and accessed from different programming languages. In this section you will find sample code that show the capabilities of the new AuthenticView and AuthenticRange interfaces. These samples should make it easier for you to get started with your own project. Although they are more complex then those you will find in the descriptions of the various methods and properties, they are fragments and might require additional code or small adaptations in order for them to be properly executed in your environment.

The listed code samples are:

- JScript: Bubble Sort Dynamic Tables
- VBScript: Using object-level events

This section also briefly describes some operations that can be performed within Authentic. This is to give you an idea of what is possible with the XMLSpy API.

4.2.1 JScript: Bubble Sort Dynamic Tables

The following JScript snippet will sort any dynamic table by the table column identified by the current cursor position. The sort process is performed on screen. The undo buffer is available for all performed operations.

If you can run JScript on you computer - as you most likely will - copy the following code into a file with extension 'js'. Execute the script by double-clicking it in Windows Explorer, or run it from the command line.

```
// some useful XMLSpy enum constants
var spyAuthenticTag = 6;
var spyAuthenticTable = 9;
var spyAuthenticTableRow = 10;
var spyAuthenticTableColumn = 11;

var spyAuthenticRangeBegin = 2;

// example call for the sort table function
try
{
  var objSpy = objSpy = WScript.GetObject("", "XMLSpy.Application");
  // use current selection to indicate which column to sort
  SortCurrentTable (objSpy.ActiveDocument.AuthenticView.Selection);
}
catch (err)
  { WScript.Echo ("Please open a document in authentic view, and select a
  table column\n" +
                  "Error : (" + (err.number & 0xffff) + ")" +
err.description); }

// we assume that XMLSpy is running, a document with a dynamic table
// is open, and the cursor is in a table column that will
// be used for sorting.
function SortCurrentTable (objCursor)
{
  if (objCursor.IsInDynamicTable())
  {
    // calculate current column index
    var nColIndex = 0;
    while (true)
    {
```

 © 2005 Altova GmbH

```
      // go left column-by-column
      try { objCursor.GotoPrevious(spyAuthenticTableColumn); }
      catch (err) { break; }
      nColIndex++;
    }

    // count number of table rows, so the bubble loops become simpler.
    // goto begin of table
    var objTableStart =
    objCursor.ExpandTo(spyAuthenticTable).CollapsToBegin().Clone();
    var nRows = 1;
    while (true)
    {
      // go down row-by-row
      try { objTableStart.GotoNext(spyAuthenticTableRow); }
      catch (err) { break; }
      nRows++;
    }

    // bubble sort through table
    for (var i = 0; i < nRows - 1; i++)
    {
      // select correct column in first table row
      var objBubble =
      objCursor.ExpandTo(spyAuthenticTable).CollapsToBegin().Clone();
      objBubble.Goto (spyAuthenticTableColumn, nColIndex,
      spyAuthenticRangeBegin).ExpandTo(spyAuthenticTag);

      // bubble this row down as far as necessary
      for (var j = 0; j < nRows - i - 1; j++)
      {
        var strField1 = objBubble.Text;
        // now look for the comparison table cell: start of next row and right
        of the correct column
        var strField2 = objBubble.GotoNext(spyAuthenticTableRow).
                                  Goto (spyAuthenticTableColumn, nColIndex,
        spyAuthenticRangeBegin).
                                  ExpandTo(spyAuthenticTag).Text;
        if (strField1 > strField2)
        {
          objBubble.MoveRowUp();  // swap the rows
          // and re-calculate objBubble to select the cell to bubble
          objBubble.GotoNext(spyAuthenticTableRow).
                    Goto (spyAuthenticTableColumn, nColIndex,
          spyAuthenticRangeBegin).
                    ExpandTo(spyAuthenticTag);
        }
      }
    }
  }
  else
    WScript.Echo ("please, select a table cell first");
}
```

4.2.2 VBScript: Using object-level events

Authentic view now supports event connection on a per-object basis. Implementation of this feature is based on COM connection points and is available in environments that support this mechanism.

The following example is a VB code snippet that shows how to connect to object-level events from within a VB project.

```
' ------------------------------------------------------------------------
```

```
--
' VB code snippet - connecting to object level events
' -------------------------------------------------------------------------
--
Dim objSpy As XMLSpyLib.Application
' use VBA keyword WithEvents to automatically connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView

' this is the event callback routine that will be connected to the
' OnSelectionChanged event of object objView
Private Sub objView_OnSelectionChanged (ByVal i_ipNewRange As
XMLSpyLib.IAuthenticRange)
    MsgBox ("new selection: " & i_ipNewRange.Text)
End Sub

' this is the event callback routine that will be connected to the
' OnSelectionChanged event of object objView
Private Sub objView_OnSelectionChanged (ByVal i_ipNewRange As
XMLSpyLib.IAuthenticRange)
    MsgBox ("new selection: " & i_ipNewRange.Text)
End Sub

' this is the event callback routine that will be connected to the
' OnMouseEvent event of object objView.
' If you click with the left mouse button while pressing a control key,
' the current selection will be set to the tag below the current
' mouse cursor position
Private Function objView_OnMouseEvent(ByVal i_nXPos As Long, ByVal i_nYPos
As Long, ByVal i_eMouseEvent As XMLSpyLib.SPYMouseEvent, ByVal i_pRange As
XMLSpyLib.IAuthenticRange) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    End If
End Function

' connect to running XMLSpy application
' this code will most likely be in some Form_Load() subroutine
Set objSpy = GetObject("", "XMLSpy.Application")
objSpy.ShowApplication (True)
If (objSpy.ActiveDocument Is Nothing) Then
    Dim objDoc As XMLSpyLib.Document
    Rem replace path below With a valid absolute path On your machine
    Set objDoc = objSpy.Documents.OpenFile("InputData\orgchart.xml", False)
End If

If (objSpy.ActiveDocument.CurrentViewMode <> spyViewContent) Then
    objSpy.ActiveDocument.SwitchViewMode (spyViewContent)
End If
Set objView = objSpy.ActiveDocument.AuthenticView
' now the object objView is set and selection change events are received

' continue here with something useful ...
' and serve the windows message loop

' if you want to stop receiving events from object objView
' add the following line somewhere appropriate
Set objView = Nothing
```


© 2005 Altova GmbH

4.3 Interfaces

Object Hierarchy

```
Application
  SpyProject
    SpyProjectItems
      SpyProjectItem
  Documents
    Document
      GridView
      AuthenticView
        AuthenticRange
        AuthenticDataTransfer (previously DocEditDataTransfer)
      OldAuthenticView (previously DocEditView, now obsolete, superseded by
      AuthenticView and AuthenticRange)
        AuthenticSelection (previously DocEditSelection, now obsolete,
        superseded by AuthenticRange)
        AuthenticEvent (previously DocEditEvent, now obsolete)
        AuthenticDataTransfer (previously DocEditDataTransfer)
      XMLData
  Dialogs
    CodeGeneratorDlg
    FileSelectionDlg
    SchemaDocumentationDlg
  DatabaseConnection
  ExportSettings
  TextImportExportSettings
  ElementList
    ElementListItem

Enumerations
```

Description

This chapter contains the reference of the XMLSpy 2005 1.2 Type Library.

Most of the given examples are written in VisualBasic. These code snippets assume that there is a variable defined and set, called **objSpy of type Application**. There are also some code samples written in JavaScript.

4.3.1 Application

See also

Methods

```
GetDatabaseImportElementList
GetDatabaseSettings
GetDatabaseTables
ImportFromDatabase

GetTextImportElementList
GetTextImportExportSettings
ImportFromText

ImportFromWord

ImportFromSchema

GetExportSettings
```

```
NewProject
OpenProject

AddMacroMenuItem
ClearMacroMenu

ShowForm

ShowApplication

URLDelete
URLMakeDirectory

Quit
```

Properties

```
Application
Parent

ActiveDocument
Documents

CurrentProject

Dialogs

WarningNumber
WarningText
```

Description

Application is the root for all other objects. It is the only object you can create by `CreateObject` (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

Events

OnBeforeOpenDocument

See also

Event: `OnBeforeOpenDocument(`*`objDialog`* as `FileSelectionDlg)`

Description

This event gets fired whenever a document gets opened via the OpenFile or OpenURL menu command. It is sent after a document file has been selected but before the document gets opened. The file selection dialog object is initialized with the name of the selected document file. You can modify this selection. To continue the opening of the document leave the `FileSelectionDlg.DialogAction` property of *io_objDialog* at its default value `spyDialogOK`. To abort the opening of the document set this property to `spyDialogCancel`.

Examples

Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

© 2005 Altova GmbH

```
Function On_BeforeOpenDocument(objDialog)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeOpenDocument(objDialog)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (26, ...)  // nEventId = 26
```

OnBeforeOpenProject

See also

Event: `OnBeforeOpenProject(objDialog as FileSelectionDlg)`

Description
This event gets fired after a project file has been selected but before the project gets opened. The file selection dialog object is initialized with the name of the selected project file. You can modify this selection. To continue the opening of the project leave the `FileSelectionDlg.DialogAction` property of *io_objDialog* at its default value `spyDialogOK`. To abort the opening of the project set this property to `spyDialogCancel`.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_BeforeOpenProject(objDialog)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeOpenProject(objDialog)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (25, ...)  // nEventId = 25
```

OnDocumentOpened

See also

Event: `OnDocumentOpened(objDocument as Document)`

Description
This event gets fired whenever a document opens in XMLSpy 2005. This can happen due to opening a file with the OpenFile or OpenURL dialog, creating a new file or dropping a file onto XMLSpy 2005. The new document gets passed as parameter. The operation cannot be canceled.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_OpenDocument(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_OpenDocument(objDocument)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (7, ...)   // nEventId = 7
```

OnProjectOpened

See also

Event: `OnProjectOpened(objProject as SpyProject)`

Description
This event gets fired whenever a project gets opened in XMLSpy 2005. The new project gets passed as parameter.

Examples
Given below are examples of how this event can be scripted.

XMLSpy scripting environment - VBScript:

```
Function On_OpenProject(objProject)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_OpenProject(objProject)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (6, ...)   // nEventId = 6
```

ActiveDocument

See also

Property: `ActiveDocument` as `Document`

Description
Reference to the active document. If no document is open, `ActiveDocument` is null (nothing).

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

© 2005 Altova GmbH

AddMacroMenuItem

See also

Method: `AddMacroMenuItem(strMacro as String,strDisplayText as String)`

Return Value

Description
Adds an menu item to the **Tools** menu. This new menu item invokes the macro defined by `strMacro`. See also Calling macros from XMLSpy 2005".

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1108	Number of macro items is limited with 16 items.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

ClearMacroMenu

See also

Method: `ClearMacroMenu()`

Return Value
None

Description
Removes all menu items from the **Tools** menu. See also Calling macros from XMLSpy 2005".

Errors

1111	The application object is no longer valid.

CurrentProject

See also

Property: `CurrentProject` as `SpyProject`

Description
Reference to the active document. If no project is open, `CurrentProject` is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Dialogs

See also

Property: `Dialogs` as `Dialogs` (read-only)

Description
Access the built-in dialogs of XMLSpy 2005.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

Documents

See also

Property: `Documents` as `Documents`

Description
Collection of all open documents. See also Simple document access.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

GetDatabaseImportElementList

See also

Method: `GetDatabaseImportElementList(`*`pImportSettings`* as `DatabaseConnection)` as `ElementList`

Description
The function returns a collection of `ElementListItems` where the properties `ElementListItem.Name` contain the names of the fields that can be selected for import and the properties `ElementListItem.ElementKind` are initialized either to *spyXMLDataAttr* or *spyXMLDataElement*, depending on the value passed in `DatabaseConnection.AsAttributes`. This list serves as a filter to what finally gets imported by a future call to `ImportFromDatabase`. Use `ElementList.RemoveElement` to exclude fields from import.

Properties mandatory to be filled out for the database connection are one of

 © 2005 Altova GmbH

`DatabaseConnection.File`, `DatabaseConnection.ADOConnection` and `DatabaseConnection.ODBCConnection`, as well as `DatabaseConnection.SQLSelect`. Use the property `DatabaseConnection.AsAttributes` to initialize `ElementListItem.ElementKind` of the resulting element list to either *spyXMLDataAttr* or *spyXMLDataElement*, respectively.

Example
See example at `ImportFromDatabase`.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.
1114	Select statement is missing.

GetDatabaseSettings

See also

Method: `GetDatabaseSettings`() as `DatabaseConnection`

Description
`GetDatabaseSettings` creates a new object of database settings. The object is used to specify database connection parameters for the methods `GetDatabaseTables`, `GetDatabaseImportElementList`, `ImportFromDatabase`, `ImportFromSchema` and `ExportToDatabase`.

Example
See example of `ImportFromDatabase`.

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

GetDatabaseTables

See also

Method: `GetDatabaseTables(`*`pImportSettings`* as `DatabaseConnection)` as `ElementList`

Description
`GetDatabaseTables` reads the table names from the database specified in *pImportSettings*. Properties mandatory to be filled out for the database connection are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` and `DatabaseConnection.ODBCConnection`. All other properties are ignored.
The function returns a collection of `ElementListItems` where the properties `ElementListItem.Name` contain the names of tables stored in the specified database. The remaining properties of `ElementListItem` are unused.

Errors

1111 The application object is no longer valid.

1100 Invalid parameter or invalid address for the return parameter was specified.

1112 Invalid database specified.

1113 Error while reading database table information.

Example

```
Dim objImpSettings As DatabaseConnection
Set objImpSettings = objSpy.GetDatabaseSettings
objImpSettings.ADOConnection = TxtADO.Text

'store table names in list box
ListTables.Clear

Dim objList As ElementList
Dim objItem As ElementListItem
On Error GoTo ErrorHandler
Set objList = objSpy.GetDatabaseTables(objImpSettings)

For Each objItem In objList
 ListTables.AddItem objItem.Name
Next
```

GetExportSettings

See also

Method: `GetExportSettings()as ExportSettings` (read-only)

Description

`GetExportSettings` creates a new object of common export settings. This object is used to pass the parameters to the export functions and defines the behaviour of the export calls. See also the export functions from `Document` and the examples at `Import and Export`.

Errors

1111 The application object is no longer valid.

1100 Invalid address for the return parameter was specified.

GetTextImportElementList

See also

Method: `GetTextImportElementList(pImportSettings as TextImportExportSettings) as ElementList`

Description

`GetTextImportElementList` retrieves importing information about the text-file as specified in `pImportSettings`. The function returns a collection of `ElementListItems` where the properties `ElementListItem.Name` contain the names of the fields found in the file. The values of remaining properties are undefined.

If the text-file does not contain a column header, set `pImportSettings.`HeaderRow to `false`. The resulting element list will contain general column names like 'Field1' and so on.

© 2005 Altova GmbH

See also `Import and export of data`.

Errors

1111 The application object is no longer valid.

1100 Invalid parameter or invalid address for the return parameter was specified.

1107 Import from database failed.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - import selected fields from text file
' ----------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = "C:\ImportMe.txt"
objImpSettings.HeaderRow = False

Dim objList As ElementList
Set objList = objSpy.GetTextImportElementList(objImpSettings)

'exclude first column
objList.RemoveItem 1

Dim objImpDoc As Document
On Error Resume Next
Set objImpDoc = objSpy.ImportFromText(objImpSettings, objList)
CheckForError
```

GetTextImportExportSettings

See also

Method: `GetTextImportExportSettings`() as `TextImportExportSettings` (read-only)

Description

`GetTextImportExportSettings` creates a new object of common import and export settings for text files. See also the example for `Application.GetTextImportElementList` and `Import and Export`.

See also `Import and export of data`.

Errors

1111 The application object is no longer valid.

1100 Invalid address for the return parameter was specified.

ImportFromDatabase

See also

Method: `ImportFromDatabase(pImportSettings` as `DatabaseConnection,pElementList` as `ElementList)` as `Document`

Return Value

Creates a new document containing the data imported from the database.

Description
`ImportFromDatabase` imports data from a database as specified in `pImportSettings` and creates a new document containing the data imported from the database. Properties mandatory to be filled out are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` or `DatabaseConnection.ODBCConnection` and `DatabaseConnection.SQLSelect`. Additionally, you can use `DatabaseConnection.AsAttributes`, `DatabaseConnection.ExcludeKeys`, `DatabaseConnection.IncludeEmptyElements` and `NumberDateTimeFormat` to further parameterize import.

The parameter `pElementList` specifies which fields of the selected data gets written into the newly created document, and which are created as elements and which as attributes. This parameter can be NULL, specifying that all selected fields will be imported as XML elements.

See `GetDatabaseSettings` and `GetDatabaseImportElementList` for necessary steps preceding any import of data from a database.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.
1114	Select statement is missing.

Example

```
 Dim objImpSettings As DatabaseConnection
 Set objImpSettings = objSpy.GetDatabaseSettings

 objImpSettings.ADOConnection = strADOConnection
 objImpSettings.SQLSelect = "SELECT * FROM MyTable"

 Dim objDoc As Document
 On Error Resume Next
 Set objDoc = objSpy.ImportFromDatabase(objImpSettings,
objSpy.GetDatabaseImportElementList(objImpSettings))
 ' CheckForError here
```

ImportFromSchema

See also

Method: `ImportFromSchema(`*`pImportSettings`* as `DatabaseConnection,`*`strTable`* as String,*`pSchemaDoc`* as `Document)` as `Document`

Return Value
Creates a new document filled with data from the specified database as specified by the schema definition in *pSchemaDoc*.

Description
`ImportFromSchema` imports data from a database specified in `pImportSettings`. Properties mandatory to be filled out are one of `DatabaseConnection.File`, `DatabaseConnection.ADOConnection` or `DatabaseConnection.ODBCConnection`. Additionally, you can use `DatabaseConnection.AsAttributes`, `DatabaseConnection.ExcludeKeys` and `NumberDateTimeFormat` to further

 © 2005 Altova GmbH

parameterize import. All other properties get ignored.

`ImportFromSchema` does not use and explicit SQL statement to select the data. Instead, it expects a structure definition of the document to create in form of an XML schema document in *pSchemaDoc*. From this definition the database select statement is automatically deduced. Specify in *strTable* the table name of the import root that will become the root node in the new document.

See `GetDatabaseSettings` and `GetDatabaseTables` for necessary steps preceding an import from a database based on a schema definition. To create the schema definition file use command 'create database schema' from the 'convert' menu of XMLSpy 2005.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from database failed.
1112	Invalid database specified.

ImportFromText

See also

Method: `ImportFromText(`*`pImportSettings`* as `TextImportExportSettings,`*`pElementList`* as `ElementList)` as `Document`

Description

`ImportFromText` imports the text file as specified in `pImportSettings`. The parameter `pElementList` can be used as import filter. Either pass the list returned by a previous call to GetTextImportElementList or `null` to import all columns. To avoid import of unnecessary columns use ElementList.RemoveElement to remove the corresponding field names from `pElementList` before calling `ImportFromText`.
The method returns the newly created document containing the imported data. This document is the same as the active document of XMLSpy 2005.

See also `Import and export of data`.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1107	Import from text file failed.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - import from text file
' ----------------------------------------------------------
Dim objImpSettings As TextImportExportSettings
Set objImpSettings = objSpy.GetTextImportExportSettings

objImpSettings.ImportFile = strFileName
objImpSettings.HeaderRow = False

Dim objImpDoc As Document
On Error Resume Next
```

```
Set objImpDoc = objSpy.ImportFromText(objImpSettings,
     objSpy.GetTextImportElementList(objImpSettings))

CheckForError
```

ImportFromWord

See also

Method: `ImportFromWord(strFile as String) as Document`

Description
`ImportFromWord` imports the MS-Word Document `strFile` into a new XML document.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified. Import from document failed.

NewProject

See also

Method: `NewProject(strPath as String,bDiscardCurrent as Boolean)`

Description
`NewProject` creates a new project.

If there is already a project open that has been modified and `bDiscardCurrent` is false, then `NewProject()` fails.

Errors

1111	The application object is no longer valid.
1102	A project is already open but *bDiscardCurrent* is *true*.
1103	Creation of new project failed.

OpenProject

See also

Method: `OpenProject(strPath as String,bDiscardCurrent as Boolean,bDialog as Boolean)`

Parameters
`strPath`
Path and file name of the project to open. Can be empty if `bDialog` is true.

`bDiscardCurrent`
Discard currently open project and possible lose changes.

`bDialog`
Show dialogs for user input.

© 2005 Altova GmbH

Return Value
None

Description
`OpenProject` opens an existing project. If there is already a project open that has been modified and `bDiscardCurrent` is false, then `OpenProject()` fails.

Errors

1111	The application object is no longer valid.
1100	Invalid parameter or invalid address for the return parameter was specified.
1101	Cannot open specified project.
1102	A project is already open but *bDiscardCurrent* is *true*.

Parent

See also

Property: `Parent` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

1111	The application object is no longer valid.
1100	Invalid address for the return parameter was specified.

Quit

See also

Method: Quit()

Return Value
None

Description
This method terminates XMLSpy 2005. All modified documents will be closed without saving the changes. This is also true for an open project.

If XMLSpy 2005 was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors

1111	The application object is no longer valid.

ShowApplication

See also

Method: `ShowApplication(bShow as Boolean)`

Return Value
None

Description
The method shows `(bShow = True)` or hides `(bShow = False)` XMLSpy 2005.

Errors
1110 The application object is no longer valid.

ShowForm

See also

Method: `ShowForm(strFormName as String) as Long`

Return Value
Returns zero if the user pressed a Cancel button or the form calls `TheView.Cancel()`.

Description
Displays the form `strFormName`.

Forms, event handlers and macros can be created with the Scripting Environment. Select "Switch to scripting environment" from the **Tools** menu to invoke the Scripting Environment.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter or invalid address for the return parameter was specified.

URLDelete

See also

Method: `URLDelete(strURL as String,strUser as String,strPassword as String)`

Return Value
None

Description
The method deletes the file at the URL `strURL`.

Errors
1111 The application object is no longer valid.
1109 Error deleting file at specified URL.

URLMakeDirectory

See also

Method: `URLMakeDirectory(strURL as String,strUser as String,strPassword as String)`

Return Value

© 2005 Altova GmbH

None

Description
The method creates a new directory at the URL `strURL`.

Errors
1111 The application object is no longer valid.
1100 Invalid parameter specified.

Visible

See also

Property: `Visible` as `Boolean`

Description
Sets or gets the visibility attribute of XMLSpy 2005. This standard automation property makes usage of `ShowApplication` obsolete.

Errors
1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningNumber

See also

Property: `WarningNumber` as integer

Description
Some methods fill the property `WarningNumber` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

WarningText

See also

Property: `WarningText` as String

Description
Some methods fill the property `WarningText` with additional information if an error occurs.

Currently just `Documents.OpenFile` fills this property.

Errors

1111 The application object is no longer valid.

1100 Invalid address for the return parameter was specified.

4.3.2 AuthenticDataTransfer

Renamed from DocEditDataTransfer to AuthenticDataTransfer

The `DocEditView` object is renamed to `OldAuthenticView`. `DocEditSelection` is renamed to `AuthenticSelection`. `DocEditEvent` is renamed to `AuthenticEvent`. `DocEditDataTransfer` is renamed to `AuthenticDataTransfer`.

Their usage—except for `AuthenticDataTransfer`—is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in `DocEditView`, `DocEditSelection`, `DocEditEvent` and `DocEditDataTransfer` is now available via `AuthenticView`, `AuthenticRange` and `AuthenticDataTransfer`. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

`getData`

Properties

`dropEffect`
`ownDrag`
`type`

Description

The events `OnDragOver` and `OnBeforeDrop` provide information about the object being dragged with an instance of type `AuthenticDataTransfer`. It contains a description of the dragged object and its content. The latter is available either as string or a pointer to a COM object supporting the `IUnkown` interface.

dropEffect

See also

Property: `dropEffect` as long

Description
The property stores the drop effect from the default event handler. You can set the drop effect if you change this value and return TRUE for the event handler (or set `AuthenticEvent.cancelBubble` to `TRUE` if you are still using the now obsolete `AuthenticEvent` interface).

© 2005 Altova GmbH

Errors
2101 Invalid address for the return parameter was specified.

getData

See also

Method: `getData()` as Variant

Description
Retrieve the data associated with the dragged object. Depending on `AuthenticDataTransfer.type`, that data is either a string or a COM interface pointer of type `IUnknown`.

Errors
2101 Invalid address for the return parameter was specified.

ownDrag

See also

Property: `ownDrag as` Boolean (read-only)

Description
The property is `TRUE` if the current dragging source comes from inside Authentic.

Errors
2101 Invalid address for the return parameter was specified.

type

See also

Property: `type` as String (read-only)

Description
Holds the type of data you get with the `DocEditDataTransfer.getData` method.

Currently supported data types are:

OWN	data from Authentic itself
TEXT	plain text
UNICODETEXT	plain text as UNICODE

Errors
2101 Invalid address for the return parameter was specified.

4.3.3 AuthenticRange

See also

The first table lists the properties and methods of `AuthenticRange` that can be used to navigate through the document and select specific portions.

Properties		Methods
`Application`	`Clone`	`MoveBegin`
`FirstTextPosition`	`CollapsToBegin`	`MoveEnd`
`FirstXMLData`	`CollapsToEnd`	`NextCursorPosition`
`FirstXMLDataOffset`	`ExpandTo`	`PreviousCursorPosition`
`LastTextPosition`	`Goto`	`Select`
`LastXMLData`	`GotoNext`	`SelectNext`
`LastXMLDataOffset`	`GotoPrevious`	`SelectPrevious`
`Parent`	`IsEmpty`	`SetFromRange`
	`IsEqual`	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
`Text`	`Copy`	`AppendRow`
	`Cut`	`DeleteRow`
	`Delete`	`DuplicateRow`
	`IsCopyEnabled`	`InsertRow`
	`IsCutEnabled`	`IsFirstRow`
	`IsDeleteEnabled`	`IsInDynamicTable`
	`IsPasteEnabled`	`IsLastRow`
	`Paste`	`MoveRowDown`
		`MoveRowUp`

The following methods provide the functionality of the Authentic entry helper windows for range objects.

© 2005 Altova GmbH

Operations of the entry helper windows

Elements	Attributes	Entities
CanPerformActionWith	GetElementAttributeValue	GetEntityNames
CanPerformAction	GetElementAttributeNames	InsertEntity
PerformAction	GetElementHierarchy	
	HasElementAttribute	
	IsTextStateApplied	
	SetElementAttributeValue	

Description
AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read/only properties AuthenticView.DocumentBegin, AuthenticView.DocumentEnd, and AuthenticView.WholeDocument. The most flexible method is AuthenticView.Goto, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above, with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition and LastTextPosition. This method requires complex internal calculations and should be used with care.
- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData, FirstXMLDataOffset, LastXMLData, and LastXMLDataOffset. This method is very efficient but requires knowledge on the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut, Copy, Paste and Delete.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData objects of the underlying document to modify them directly.

AppendRow

See also

Method: `AppendRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Append row at end of current dynamically growable table
' ----------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.AppendRow
     ' objRange points to beginning of new row
     objRange.Select
End If
```

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

CanPerformAction

See also

Method: `CanPerformAction` (*`eAction`* as `SPYAuthenticActions`, *`strElementName`* as String) as Boolean

Description
`CanPerformAction` and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the

© 2005 Altova GmbH

range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use `CanPerformActionWith`.

Errors

2001 The authentic range object or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

2007 Invalid action was specified.

Examples

See `PerformAction`.

CanPerformActionWith

See also

Method: `PerformActionWith` (*`eAction`* as `SPYAuthenticActions`, *`out_arrElementNames`* as Variant)

Description

`PerformActionWith` and its related methods, enable access to the entry-helper functions of Authentic. These function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use `CanPerformActionWith`.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

2007 Invalid action was specified.

Examples

See `PerformAction`.

Clone

See also

Method: `Clone()` as `AuthenticRange`

Description

Returns a copy of the range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

CollapsToBegin

See also

Method: `CollapsToBegin()` as `AuthenticRange`

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

CollapsToEnd

See also

Method: `CollapsToEnd()` as `AuthenticRange`

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Copy

See also

Method: `Copy()` as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Cut

See also

Method: `Cut()` as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.

© 2005 Altova GmbH

Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Delete

See also

Method: `Delete()` as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors

2001 The authentic range object or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

DeleteRow

See also

Method: `DeleteRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Delete selected row from dynamically growing table
' ----------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we are in a table
If objRange.IsInDynamicTable Then
     objRange.DeleteRow
End If
```

DuplicateRow

See also

Method: `DuplicateRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------------------
' XMLSpy scripting environment - VBScript
' duplicate row in current dynamically growable table
' ---------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.DuplicateRow
     ' objRange points to beginning of new row
     objRange.Select
End If
```

ExpandTo

See also

Method: `ExpandTo` (*`eKind`* as `SPYAuthenticElementKind`), as `AuthenticRange`

Description
Selects the whole element of type `eKind`, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Range expansion would be beyond end of document.
2005	Invalid address for the return parameter was specified.

FirstTextPosition

See also

Property: `FirstTextPosition` as Long

Description
Set or get the left-most text position index of the range object. This index is always less or equal

 © 2005 Altova GmbH

to `LastTextPosition`. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set `FirstTextPosition` to a value greater than the current `LastTextPosition`, `LastTextPosition` gets set to the new `FirstTextPosition`.

HINT: Use text cursor positions with care, since this is a costly operation compared to `XMLData` based cursor positioning.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2006	A text position outside the document was specified.

Examples

```
  ' ---------------------------------------
  ' XMLSpy scripting environment - VBScript
  ' ---------------------------------------
  Dim objAuthenticView
  ' we assume that the active document is open in authentic view mode
  Set objAuthenticView = Application.ActiveDocument.AuthenticView

  nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
  nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

  ' let's create a range that selects the whole document
  ' in an inefficient way
  Dim objRange
  ' we need to get a (any) range object first
  Set objRange = objAuthenticView.DocumentBegin
  objRange.FirstTextPosition = nDocStartPosition
  objRange.LastTextPosition = nDocEndPosition

  ' let's check if we got it right
  If objRange.isEqual(objAuthenticView.WholeDocument) Then
       MsgBox "Test using direct text cursor positioning was ok"
  Else
       MsgBox "Ooops!"
End If
```

FirstXMLData

See also

Property: `FirstXMLData` as `XMLData`

Description

Set or get the first `XMLData` element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the `FirstXMLDataOffset` attribute.

Whenever you set `FirstXMLData` to a new data object, `FirstXMLDataOffset` gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set `FirstXMLData` / `FirstXMLDataOffset` selects a position greater then the current `LastXMLData` / `LastXMLDataOffset`, the latter gets moved to the new start position.

HINT: You can use the FirstXMLData and LastXMLData properties, to directly access and manipulate the underlying XML document in those cases where the methods available with the AuthenticRange object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.

2005 Invalid address for the return parameter was specified.

2008 Internal error

2009 The XMLData object cannot be accessed.

Examples

```
' ------------------------------------------------
' XMLSpy scripting environment - VBScript
' show name of currently selected XMLData element
' ------------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXMLData = objAuthenticView.Selection.FirstXMLData
' authentic view adds a 'text' child element to elements
' of the document which have content. So we have to go one
' element up.
Set objXMLData = objXMLData.Parent
MsgBox "Current selection selects element " & objXMLData.Name
```

FirstXMLDataOffset

See also

Property: FirstXMLDataOffset as Long

Description

Set or get the cursor position offset inside FirstXMLData element for the beginning of the range. Offset positions are based on the characters returned by the Text property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If FirstXMLData / FirstXMLDataOffset selects a position after the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

Errors

2001 The authentic range object, or its related view object is not valid.

2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

 © 2005 Altova GmbH

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
     objRange.Select()
Else
     MsgBox "Oops"
End If
```

GetElementAttributeNames

See also

Method: `GetElementAttributeNames` (*`strElementName`* as String, *`out_arrAttributeNames`* as Variant)

Description
Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods `GetElementAttributeValue` and `SetElementAttributeValue`.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid element name was specified. Invalid address for the return parameter was specified.

Examples
See `SetElementAttributeValue`.

GetElementAttributeValue

See also

Method: `GetElementAttributeValue` (*`strElementName`* as String, *`strAttributeName`* as String) as String

Description
Retrieve the value of the attribute specified in `strAttributeName`, for the element identified with `strElementName`. If the attribute is supported but has no value assigned, the empty string

is returned. To find out the names of attributes supported by an element, use `GetElementAttributeNames`, or `HasElementAttribute`.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid address for the return parameter was specified.

Examples

See `SetElementAttributeValue`.

GetElementHierarchy

See also

Method: `GetElementHierarchy` (*`out_arrElementNames`* as Variant)

Description

Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single `XMLData` element.

The names of the element hierarchy, together with the range object uniquely identify `XMLData` elements in the document. The attributes of these elements can be directly accessed by `GetElementAttributeNames`, and related methods.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Examples

See `SetElementAttributeValue`.

GetEntityNames

See also

Method: `GetEntityNames` (*`out_arrEntityNames`* as Variant)

Description

Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the `InsertEntity` function.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Examples

See `InsertEntity`.

© 2005 Altova GmbH

Goto

See also

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition) as AuthenticRange

Description
Sets the range to point to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom.

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid start position specified. Invalid address for the return parameter was specified.

GotoNext

See also

Method: GotoNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Sets the range to the beginning of the next element of type eKind. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document word-by-word
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
     objRange.GotoNext(spyAuthenticWord).Select
```

```
      If ((Err.number - vbObjecterror) = 2003) Then
            bEndOfDocument = True
            Err.Clear
      ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
      End If
  Wend
```

GotoNextCursorPosition

See also

Method: `GotoNextCursorPosition()` as `AuthenticRange`

Description
Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid address for the return parameter was specified.

GotoPrevious

See also

Method: `GotoPrevious` (*`eKind`* as `SPYAuthenticElementKind`) as `AuthenticRange`

Description
Sets the range to the beginning of the element of type `eKind` which is before the beginning of the current range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
  ' ----------------------------------------
  ' XMLSpy scripting environment - VBScript
  ' Scan through the whole document tag-by-tag
  ' ----------------------------------------
  Dim objAuthenticView
  ' we assume that the active document is open in authentic view mode
  Set objAuthenticView = Application.ActiveDocument.AuthenticView

  Dim objRange
  Set objRange = objAuthenticView.DocumentEnd
  Dim bEndOfDocument
  bBeginOfDocument = False

  On Error Resume Next
  While Not bBeginOfDocument
       objRange.GotoPrevious(spyAuthenticTag).Select
       If ((Err.number - vbObjecterror) = 2004) Then
```


© 2005 Altova GmbH

```
            bBeginOfDocument = True
            Err.Clear
      ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
      End If
  Wend
```

GotoPreviousCursorPosition

See also

Method: `GotoPreviousCursorPosition()` as `AuthenticRange`

Description
Set the range to the cursor position immediately before the current position. Returns the modified object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.
2005	Invalid address for the return parameter was specified.

HasElementAttribute

See also

Method: `HasElementAttribute` (*`strElementName`* as String, *`strAttributeName`* as String) as Boolean

Description
Tests if the enclosing element with name `strElementName`, supports the attribute specified in `strAttributeName`.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid element name was specified. Invalid address for the return parameter was specified.

InsertEntity

See also

Method: `InsertEntity` (*`strEntityName`* as String)

Description
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by `GetEntityNames`.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Unknown entry name was specified.

Examples

```
' ------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert the first entity in the list of available entities
' ------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we get the names of all available entities as they
' are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

' we insert the first one of the list
If UBound(arrEntities) >= 0 Then
     objRange.InsertEntity arrEntities(0)
     objRange.Select()
Else
     MsgBox "Sorry, no entities are available for this document"
End If
```

InsertRow

See also

Method: `InsertRow()` as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range, gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
     objRange.InsertRow
     ' objRange points to beginning of new row
     objRange.Select
End If
```

IsCopyEnabled

See also

Property: `IsCopyEnabled` as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

 © 2005 Altova GmbH

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsCutEnabled

See also

Property: `IsCutEnabled` as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsDeleteEnabled

See also

Property: `IsDeleteEnabled` as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsEmpty

See also

Method: `IsEmpty()` as Boolean

Description
Tests if the first and last position of the range are equal.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsEqual

See also

Method: `IsEqual` (*`objCmpRange`* as `AuthenticRange`) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors

2001 One of the two range objects being compared, is invalid.

2005 Invalid address for a return parameter was specified.

IsFirstRow

See also

Property: `IsFirstRow()` as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsInDynamicTable

See also

Method: `IsInDynamicTable()` as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsLastRow

See also

Property: `IsLastRow()` as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extend of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

© 2005 Altova GmbH

IsPasteEnabled

See also

Property: `IsPasteEnabled` as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

IsTextStateApplied

See also

Method: `IsTextStateApplied` (*`i_strElementName`* as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name `i_strElementName`. Common examples for the parameter `i_strElementName` are "strong", "bold" or "italic".

Errors

2001 The authentic range object, or its related view object is no longer valid.

2005 Invalid address for the return parameter was specified.

LastTextPosition

See also

Property: `LastTextPosition` as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to `FirstTextPosition`. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set `LastTextPosition` to a value less then the current `FirstTextPosition`, `FirstTextPosition` gets set to the new `LastTextPosition`.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001 The authentic range object, or its related view object is not valid.

2005 Invalid address for the return parameter was specified.

2006 A text position outside the document was specified.

Examples

```
' --------------------------------------
' XMLSpy scripting environment - VBScript
' --------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
     MsgBox "Test using direct text cursor positioning was ok"
Else
     MsgBox "Oops!"
End If
```

LastXMLData

See also

Property: `LastXMLData` as `XMLData`

Description

Set or get the last `XMLData` element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the `LastXMLDataOffset` attribute.

Whenever you set `LastXMLData` to a new data object, `LastXMLDataOffset` gets set to the last cursor position inside this element. Only `XMLData` objects that have a cursor position may be used. If you set `LastXMLData` / `LastXMLDataOffset`, select a position less then the current `FirstXMLData` / `FirstXMLDataOffset`, the latter gets moved to the new end position.

HINT: You can use the `FirstXMLData` and `LastXMLData` properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the `AuthenticRange` object are not sufficient.

Errors

 © 2005 Altova GmbH

2001 The authentic range object, or its related view object is not valid.

2005 Invalid address for the return parameter was specified.

2008 Internal error

2009 The `XMLData` object cannot be accessed.

LastXMLDataOffset

See also

Property: `LastXMLDataOffset` as Long

Description
Set or get the cursor position inside `LastXMLData` element for the end of the range.

Offset positions are based on the characters returned by the `Text` property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If `LastXMLData` / `LastXMLDataOffset` selects a position before `FirstXMLData` / `FirstXMLDataOffset`, the latter gets moved to the new end position.

Errors

2001 The authentic range object, or its related view object is not valid.

2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' --------------------------------------------
' XMLSpy scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' --------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)
```

```
' were we successful?
If objRange.IsEqual(objRange2) Then
     objRange.Select()
Else
     MsgBox "Ooops"
End If
```

MoveBegin

See also

Method: `MoveBegin` (*`eKind`* as `SPYAuthenticElementKind`, *`nCount`* as Long) as `AuthenticRange`

Description
Move the beginning of the range to the beginning of the `nCount` element of type `eKind`. Counting starts at the current beginning of the range object.

Use positive numbers for `nCount` to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

MoveEnd

See also

Method: `MoveEnd` (*`eKind`* as `SPYAuthenticElementKind`, *`nCount`* as Long) as `AuthenticRange`

Description
Move the end of the range to the begin of the `nCount` element of type `eKind`. Counting starts at the current end of the range object.

Use positive numbers for `nCount` to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

© 2005 Altova GmbH

MoveRowDown

See also

Method: `MoveRowDown()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

MoveRowUp

See also

Method: `MoveRowUp()` as Boolean

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Examples
See `JScript - Bubble Sort Dynamic Tables`.

Parent

See also

Property: `Parent` as `AuthenticView` (read-only)

Description
Access the view that owns this range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Paste

See also

Method: `Paste()` as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

PerformAction

See also

Method: `PerformAction` (*`eAction`* as `SPYAuthenticActions`, *`strElementName`* as String) as Boolean

Description
`PerformAction` and its related methods, give access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use `CanPerformActionWith`.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.
2007	Invalid action was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Insert the innermost element
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' we determine the elements that can be inserted at the current position
Dim arrElements()
objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

' we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
     objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
```

 © 2005 Altova GmbH

```
      ' objRange now points to the beginning of the inserted element
      ' we set a default value and position at its end
      objRange.Text = "Hello"
      objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
      MsgBox "Can't insert any elements at current position"
End If
```

Select

See also

Method: `Select()`

Description
Makes this range the current user interface selection. You can achieve the same result using: '*objRange.Parent.Selection = objRange*'

Errors

2001 The authentic range object or its related view object is no longer valid.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```

SelectNext

See also

Method: `SelectNext` (*`eKind`* as `SPYAuthenticElementKind`) as `AuthenticRange`

Description
Selects the element of type `eKind` after the current end of the range. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.

2003 Target lies after end of document.

2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document word-by-word
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
```

```
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
     objRange.SelectNext(spyAuthenticWord).Select
     If ((Err.number - vbObjecterror) = 2003) Then
            bEndOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```

SelectPrevious

See also

Method: `GotoPrevious` (*`eKind`* as `SPYAuthenticElementKind`) as `AuthenticRange`

Description
Selects the element of type `eKind` before the current beginning of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.
2005	Invalid element kind specified. Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bEndOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
     objRange.SelectPrevious(spyAuthenticTag).Select
     If ((Err.number - vbObjecterror) = 2004) Then
            bBeginOfDocument = True
            Err.Clear
     ElseIf (Err.number <> 0) Then
            Err.Raise  ' forward error
     End If
Wend
```


© 2005 Altova GmbH

SetElementAttributeValue

See also

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

Description
Retrieve the value of the attribute specified in `strAttributeName` for the element identified with `strElementName`. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use `GetElementAttributeNames`, or `HasElementAttribute`.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid element name was specified. Invalid attribute name was specified. Invalid attribute value was specified.

Examples

```
' ------------------------------------------
' XMLSpy scripting environment - VBScript
' Get and set element attributes
' ------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
     If UBound(arrElements) >= 0 Then
            ' we use the top level element and find out its valid attributes
           Dim arrAttrs()
           objRange.GetElementAttributeNames arrElements(0), arrAttrs

           If UBound(arrAttrs) >= 0 Then
                  ' we retrieve the current value of the first valid
attribute
                 Dim strAttrVal
                 strAttrVal = objRange.GetElementAttributeValue
(arrElements(0), arrAttrs(0))
                 msgbox "current value of " & arrElements(0) & "//" &
arrAttrs(0) & " is: " & strAttrVal

                 ' we change this value and read it again
                 strAttrVal = "Hello"
                 objRange.SetElementAttributeValue arrElements(0),
arrAttrs(0), strAttrVal
                 strAttrVal = objRange.GetElementAttributeValue
(arrElements(0), arrAttrs(0))
                 msgbox "new value of " & arrElements(0) & "//" &
arrAttrs(0) & " is: " & strAttrVal
           End If
     End If
End If
```

SetFromRange

See also

Method: `SetFromRange` (*`objSrcRange`* as `AuthenticRange`)

Description
Sets the range object to the same beginning and end positions as `objSrcRange`.

Errors

2001	One of the two range objects, is invalid.
2005	Null object was specified as source object.

Text

See also

Property: `Text` as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use `Cut`, `Delete` or `PerformAction` instead.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for a return parameter was specified.

4.3.4 AuthenticView

See also

© 2005 Altova GmbH

Properties	Methods	Events
Application	Goto	OnBeforeCopy
AsXMLString	IsRedoEnabled	OnBeforeCut
DocumentBegin	IsUndoEnabled	OnBeforeDelete
DocumentEnd	Print	OnBeforeDrop
Event	Redo	OnBeforePaste
MarkupVisibility	Undo	OnDragOver
Parent	UpdateXMLInstanceEntities	OnKeyBoardEvent
Selection		OnMouseEvent
XMLDataRoot		OnSelectionChanged
WholeDocument		

Description
AuthenticView and its child objects AuthenticRange and AuthenticDataTransfer provide you with an interface for Authentic, which allow easy and consistent modification of document contents. These interfaces replace the following interfaces which are marked now as **obsolete**:
OldAuthenticView (old name was DocEditView)
AuthenticSelection (old name was DocEditSelection, superseded by AuthenticRange)
AuthenticEvent (old name was DocEditEvent)Interfaces

AuthenticView gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

AuthenticView uses objects of type AuthenticRange to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties DocumentBegin, DocumentEnd, or WholeDocument for simple selections, while using the Goto method for more complex selections. To navigate relative to a given document range, see the methods and properties of the AuthenticRange object.

Examples

```
' --------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' --------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
     ' we have an active document, now check for view mode
     If (objDocument.CurrentViewMode <> spyViewContent) Then
           If (Not objDocument.SwitchViewMode (spyViewContent)) Then
                 MsgBox "Active document does not support authentic view
mode"
           Else
                 ' now it is safe to access the authentic view object
                 Dim objAuthenticView
                 Set objAuthenticView = objDocument.AuthenticView
                 ' now use the authentic view object

           End If
```

```
     End If
Else
     MsgBox "No document is open"
End If
```

Events

OnBeforeCopy

See also

Event: `OnBeforeCopy()` as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_DocEditCopy()
     ' On_BeforeCopy = False  ' to disable operation
End Function
```

XMLSpy scripting environment - JScript:
```
function On_DocEditCopy()
{
     // return false;  /* to disable operation */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (21, ...)  // nEventId = 21
```

Description
This event gets triggered before a copy operation gets performed on the document. Return *True* (or nothing) to allow copy operation. Return *False* to disable copying.

OnBeforeCut

See also

Event: `OnBeforeCut()` as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_DocEditCut()
     ' On_BeforeCopy = False  ' to disable operation
End Function
```

XMLSpy scripting environment - JScript:
```
function On_DocEditCut()
{
     // return false;  /* to disable operation */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (20, ...)  // nEventId = 20
```

Description
This event gets triggered before a cut operation gets performed on the document. Return *True* (or nothing) to allow cut operation. Return *False* to disable operation.

© 2005 Altova GmbH

OnBeforeDelete

See also

Event: `OnBeforeDelete()` as Boolean

XMLSpy scripting environment - VBScript:
```
Function On_DocEditClear()
     ' On_BeforeCopy = False  ' to disable operation
End Function
```

XMLSpy scripting environment - JScript:
```
function On_DocEditClear()
{
     // return false;  /* to disable operation */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (22, ...)  // nEventId = 22
```

Description
This event gets triggered before a delete operation gets performed on the document. Return *True* (or nothing) to allow delete operation. Return *False* to disable operation.

OnBeforeDrop

See also

Event: `OnBeforeDrop (i_nXPos` as Long, `i_nYPos` as Long, `i_ipRange` as `AuthenticRange, i_ipData` as `cancel`Boolean

XMLSpy scripting environment - VBScript:
```
Function On_DocEditDrop(nXPos, nYPos, objRange, objData)
     ' On_BeforeCopy = False  ' to disable operation
End Function
```

XMLSpy scripting environment - JScript:
```
function On_DocEditDrop(nXPos, nYPos, objRange, objData)
{
     // return false;  /* to disable operation */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (11, ...)  // nEventId = 11
```

Description
This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancle the drop operation. Return *True* (or nothing) to continue normal operation.

Examples

```
' ------------------------------------------------------------------------
--
' VB code snippet - connecting to object level events
' ------------------------------------------------------------------------
--
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnBeforeDrop
' event of object objView
Private Function objView_OnBeforeDrop(ByVal i_nXPos As Long, ByVal i_nYPos
As Long,
                                      ByVal i_ipRange As IAuthenticRange,
                                      ByVal i_ipData As
IAuthenticDataTransfer) As Boolean

    If (Not i_ipRange Is Nothing) Then
        MsgBox ("Dropping on content is prohibited");
        Return False;
    Else
        Return True;
    End If
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

OnBeforePaste

See also

Event: OnBeforePaste (*objData* as Variant, *strType* as String) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditPaste(objData, strType)
    ' On_BeforeCopy = False  ' to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditPaste(objData, strType)
{
    // return false;  /* to disable operation */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (19, ...)    // nEventId = 19
```

Description

This event gets triggered before a paste operation gets performed on the document. The parameter *strType* is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases *objData* contains a string representation of the object that will be pasted. In the later

 © 2005 Altova GmbH

case, *objData* contains a pointer to an IUnknown COM interface.

Return *True* (or nothing) to allow paste operation. Return *False* to disable operation.

OnDragOver

See also

Event: OnDragOver (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent, *objRange* as AuthenticRange, *objData* as AuthenticDataTransfer) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
    ' On_BeforeCopy = False  ' to disable operation
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditDragOver(nXPos, nYPos, eMouseEvent, objRange, objData)
{
    // return false;  /* to disable operation */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (10, ...)  // nEventId = 10
```

Description

This event gets triggered whenever an object from within our outside of Authentic gets dragged with the mouse over the application window. All event related information gets passed as parameters.

The first three parameters specify the mouse position, the mouse button status and the status of the virtual keys at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drag operation. Return *True* (or nothing) to continue normal operation.

Examples

```
' ------------------------------------------------------------------------
--
' VB code snippet - connecting to object level events
' ------------------------------------------------------------------------
--
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnDragOver
' event of object objView
Private Function objView_OnDragOver(ByVal i_nXPos As Long, ByVal i_nYPos As
Long,
                                    ByVal i_eMouseEvent As SPYMouseEvent,
                                    ByVal i_ipRange As IAuthenticRange,
                                    ByVal i_ipData As
IAuthenticDataTransfer) As Boolean

    If (((i_eMouseEvent And spyShiftKeyDownMask) <> 0) And
```

```
                (Not i_ipRange Is Nothing)) Then
          MsgBox ("Floating over element " &
i_ipRange.FirstXMLData.Parent.Name);
    End If

    Return True;
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

OnKeyboardEvent

See also

Event: `OnKeyboardEvent` (*`eKeyEvent`* as SPYKeyEvent, *`nKeyCode`* as Long, *`nVirtualKeyStatus`* as Long) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditKeyboardEvent(eKeyEvent, nKeyCode,
nVirtualKeyStatus)
    ' On_DocEditKeyboardEvent = True  ' to cancel bubbling of event
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditKeyboardEvent(eKeyEvent, nKeyCode,
nVirtualKeyStatus)
{
    // return false;  /* to cancel bubbling of event */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (30, ...)  // nEventId = 30
```

Description

This event gets triggered for *WM_KEYDOWN*, *WM_KEYUP* and *WM_CHAR* Windows messages.

The actual message type is available in the *eKeyEvent* parameter. The status of virtual keys is combined in the parameter *nVirtualKeyStatus*. Use the bit-masks defined in the enumeration datatype `SPYVirtualKeyMask`, to test for the different keys or their combinations.

REMARK: The following events from the scripting environment and IDE Plugin of XMLSpy 2005 are still supported but become obsolete with this event:

```
On_DocEditKeyUp()           IXMLSpyPlugIn.OnEvent (13, ...)    //
nEventId = 13
On_DocEditKeyDown()         IXMLSpyPlugIn.OnEvent (12, ...)    //
nEventId = 12
On_DocEditKeyPressed()      IXMLSpyPlugIn.OnEvent (14, ...)    //
nEventId = 14
```

Examples

```
' ---------------------------------------------------------------------
```


© 2005 Altova GmbH

```
--
' VB code snippet - connecting to object level events
' ------------------------------------------------------------------------
--
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnKeyboard
' event of object objView
Private Function objView_OnKeyboardEvent(ByVal i_keyEvent As Long, ByVal
io_pnKeyCode As Long, ByVal i_nVirtualKeyStatus As Long) As Boolean
    If ((i_keyEvent = XMLSpyLib.spyKeyUp) And ((i_nVirtualKeyStatus And
XMLSpyLib.spyCtrlKeyMask) <> 0)) Then
        MsgBox ("Ctrl " & io_pnKeyCode & " pressed")
        objView_OnKeyboardEvent = True
    Else
        objView_OnKeyboardEvent = False
    End If
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

OnMouseEvent

See also

Event: `OnMouseEvent` (*`nXPos`* as Long, *`nYPos`* as Long, *`eMouseEvent`* as `SPYMouseEvent`, *`objRange`* as `AuthenticRange`) as Boolean

XMLSpy scripting environment - VBScript:

```
Function On_DocEditMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
    ' On_DocEditMouseEvent = True  ' to cancel bubbling of event
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
    // return false;  /* to cancel bubbling of event */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (31, ...)  // nEventId = 31
```

Description

This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the *eMouseEvent* parameter. Use the bit-masks defined in the enumeration datatype `SPYMouseEvent` to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range objects always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is *null*.

REMARK: The following events from the scripting environment and IDE Plugin of XMLSpy 2005 are still supported but become obsolete with this event:

```
On_DocEditMouseMove()              IXMLSpyPlugIn.OnEvent (15, ...)    //
nEventId = 15
On_DocEditButtonUp()               IXMLSpyPlugIn.OnEvent (16, ...)    //
nEventId = 16
On_DocEditButtonDown()             IXMLSpyPlugIn.OnEvent (17, ...)    //
nEventId = 17
On_DocEditButtonDoubleClick()      IXMLSpyPlugIn.OnEvent (24, ...)    //
nEventId = 24
```

Examples

```
' ------------------------------------------------------------------------
--
' VB code snippet - connecting to object level events
' ------------------------------------------------------------------------
--
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnMouseEvent
' event of object objView. If you click with the left mouse button
' while pressing a control key, the current selection will be set
' to the tag below the current mouse cursor position
Private Function objView_OnMouseEvent(ByVal i_nXPos As Long, ByVal i_nYPos
As Long, ByVal i_eMouseEvent As XMLSpyLib.SPYMouseEvent, ByVal i_pRange As
XMLSpyLib.IAuthenticRange) As Boolean
    If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
        On Error Resume Next
        i_pRange.Select
        objView_OnMouseEvent = True
    Else
        objView_OnMouseEvent = False
    End If
End Function

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

OnSelectionChanged

See also

Event: `OnSelectionChanged (objNewSelection as AuthenticRange)`

XMLSpy scripting environment - VBScript:

```
Function On_DocEditSelectionChanged (objNewSelection)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_DocEditSelectionChanged (objNewSelection)
{
}
```


© 2005 Altova GmbH

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (23, ...)  // nEventId = 23
```

Description
This event gets triggered whenever the selection in the user interface changes.

Examples

```
' ------------------------------------------------------------------------
--
' VB code snippet - connecting to object level events
' ------------------------------------------------------------------------
--
' access XMLSpy (without checking for any errors)
Dim objSpy As XMLSpyLib.Application
Set objSpy = GetObject("", "XMLSpy.Application")

' this is the event callback routine connected to the OnSelectionChanged
' event of object objView
Private Sub objView_OnSelectionChanged (ByVal i_ipNewRange As
XMLSpyLib.IAuthenticRange)
    MsgBox ("new selection: " & i_ipNewRange.Text)
End Sub

' use VBA keyword WithEvents to connect to object-level event
Dim WithEvents objView As XMLSpyLib.AuthenticView
Set objView = objSpy.ActiveDocument.AuthenticView

' continue here with something useful ...
' and serve the windows message loop
```

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

AsXMLString

See also

Property: `AsXMLString` as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new `XMLString` does not match the actual schema file error 2011 gets returned.

Errors

2000 The authentic view object is no longer valid.

2011 `AsXMLString` was set to a value which is no valid XML for the current schema file.

DocumentBegin

See also

Property: `DocumentBegin` as `AuthenticRange` (read-only)

Description
Retrieve a range object that points to the beginning of the document.

Errors

2000 The authentic view object is no longer valid.

2005 Invalid address for the return parameter was specified.

DocumentEnd

See also

Property: `DocumentEnd` as `AuthenticRange` (read-only)

Description
Retrieve a range object that points to the end of the document.

Errors

2000 The authentic view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Event

See also

Property: `Event` as `AuthenticEvent` (read-only)

Description
This property gives access to parameters of the last event in the same way as `OldAuthenticView.event` does. Since all events for the scripting environment and external clients are now available with parameters this `Event` property should only be used from within IDE-Plugins.

Errors

2000 The authentic view object is no longer valid.

2005 Invalid address for the return parameter was specified.

Goto

See also

© 2005 Altova GmbH

Method: `Goto` (*`eKind`* as SPYAuthenticElementKind, *`nCount`* as Long, *`eFrom`* as *`SPYAuthenticDocumentPosition`*) as *`AuthenticRange`*

Description
Retrieve a range object that points to the beginning of the *`nCount`* element of type *`eKind`*. The start position is defined by the parameter *`eFrom`*. Use positive values for *`nCount`* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors

2000	The authentic view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*. Invalid address for the return parameter was specified.

Examples

```
' --------------------------------------
' XMLSpy scripting environment - VBScript
' --------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

On Error Resume Next
Dim objRange
' goto beginning of first table in document
Set objRange = objAuthenticView.Goto (spyAuthenticTable, 1,
spyAuthenticDocumentBegin)
If (Err.number = 0) Then
     objRange.Select()
Else
     MsgBox "No table found in document"
End If
```

IsRedoEnabled

See also

Property: `IsRedoEnabled` as Boolean (read-only)

Description
True if redo steps are available and `Redo` is possible.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

IsUndoEnabled

See also

Property: `IsUndoEnabled` as Boolean (read-only)

Description
True if undo steps are available and `Undo` is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

MarkupVisibility

See also

Property: `MarkupVisibility` as `SPYAuthenticMarkupVisibility`

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Document` (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Print

See also

Method: `Print` (*`bWithPreview`* as Boolean, *`bPromptUser`* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

Errors
2000 The authentic view object is no longer valid.

© 2005 Altova GmbH

Redo

See also

Method: Redo() as Boolean

Description
Redo the modification undone by the last undo command.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

Selection

See also

Property: Selection as AuthenticRange

Description
Set or get current text selection in user interface.

Errors

2000	The authentic view object is no longer valid.
2002	No cursor selection is active.
2005	Invalid address for the return parameter was specified.

Examples

```
' --------------------------------------
' XMLSpy scripting environment - VBScript
' --------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' if we are the end of the document, re-start at the beginning
If (objAuthenticView.Selection.IsEqual(objAuthenticView.DocumentEnd)) Then
     objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
     ' objAuthenticView.Selection =
objAuthenticView.Selection.GotoNextCursorPosition()
     ' or shorter:
      objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

Undo

See also

Method: Undo() as Boolean

Description
Undo the last modification of the document from within this view.

Errors

2000 The authentic view object is no longer valid.

2005 Invalid address for the return parameter was specified.

UpdateXMLInstanceEntities

See also

Method: `UpdateXMLInstanceEntities()`

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example

```
// ----------------------------------------
// XMLSpy scripting environment - JavaScript
// ----------------------------------------
if(Application.ActiveDocument && (Application.ActiveDocument.CurrentViewMode
== 4))
{
     var objDocType;
     objDocType =
Application.ActiveDocument.DocEditView.XMLRoot.GetFirstChild(10);

     if(objDocType)
     {
            var objEntity = Application.ActiveDocument.CreateChild(14);
            objEntity.Name = "child";
            objEntity.TextValue = "SYSTEM \"child.xml\"";
            objDocType.AppendChild(objEntity);

            Application.ActiveDocument.AuthenticView.UpdateXMLInstanceEntitie
s();
     }
}
```

WholeDocument

See also

Property: `WholeDocument` as `AuthenticRange` (read-only)

Description
Retrieve a range object that selects the whole document.

Errors

2000 The authentic view object is no longer valid.

2005 Invalid address for the return parameter was specified.

© 2005 Altova GmbH

XMLDataRoot

See also

Property: `XMLDataRoot` as `XMLData` (read-only)

Description
Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors

2000 The authentic view object is no longer valid.

2005 Invalid address for the return parameter was specified.

4.3.5 CodeGeneratorDlg

See also

Properties and Methods

Standard automation properties
`Application`
`Parent`

Programming language selection properties
`ProgrammingLanguage`
`TemplateFileName`

Settings for C++ code
`CPPSettings_DOMType`
`CPPSettings_LibraryType`
`CPPSettings_UseMFC`

Settings for C# code
`CSharpSettings_ProjectType`

Dialog handling for above code generation properties
`PropertySheetDialogAction`

Output path selection properties
`OutputPath`
`OutputPathDialogAction`

Presentation of result
`OutputResultDialogAction`

Description
Use this object to configure the generation of program code for schema files. The method `GenerateProgramCode` expects a CodeGeneratorDlg as parameter to configure code generation as well as the associated user interactions.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

CPPSettings_DOMType

Property: `CPPSettings_DOMType` as `SPYDOMType`

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_LibraryType

Property: `CPPSettings_LibraryType` as `SPYLibType`

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

CPPSettings_UseMFC

Property: `CPPSettings_UseMFC` as Boolean

Description
Defines one of the settings that configure generation of C++ code.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

© 2005 Altova GmbH

CSharpSettings_ProjectType

Property: `CSharpSettings_ProjectType` as `SPYProjectType`

Description
Defines the only setting to configure generation of C# code.

Errors

2200	The object is no longer valid.
2201	Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputPath

Property: `OutputPath` as String

Description
Selects the base directory for all generated code.

Errors

2200	The object is no longer valid.
2201	Invalid address for the return parameter was specified.

OutputPathDialogAction

Property: `OutputPathDialogAction` as `SPYDialogAction`

Description
Defines how the sub-dialog for selecting the code generation output path gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current value of the OutputPath property as default. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors

2200	The object is no longer valid.
2201	Invalid action passed as parameter or an invalid address was specified for the return parameter.

OutputResultDialogAction

Property: `OutputResultDialogAction` as `SPYDialogAction`

Description
Defines how the sub-dialog that asks to show the result of the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors

2200	The object is no longer valid.
2201	Invalid action passed as parameter or an invalid address was specified for the return parameter.

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

ProgrammingLanguage

Property: `ProgrammingLanguage` as `ProgrammingLanguage`

Description
Selects the output language for the code to be generated.

CAUTION: Setting this property to one of C++, C# or Java, changes the property TemplateFileName to the appropriate template file delivered with XMLSpy 2005 as well. If you want to generate C++, C# or Java code based on your own templates, set first the programming language and then select your template file.

Errors
2200 The object is no longer valid.
2201 Invalid address for the return parameter was specified.

PropertySheetDialogAction

Property: `PropertySheetDialogAction` as `SPYDialogAction`

Description
Defines how the sub-dialog that configures the code generation process gets handled. Set this value to *spyDialogUserInput(2)* to show the dialog with the current values as defaults. Use *spyDialogOK(0)* to hide the dialog from the user.

Errors
2200 The object is no longer valid.
2201 Invalid action passed as parameter or an invalid address was specified for the return parameter.

TemplateFileName

Property: `TemplateFileName` as String

Description
Selects the code generation template file. XMLSpy 2005 comes with template files for C++, C# or Java in the SPL folder of your installation directory.

Setting this property to one of the code generation template files of your XMLSpy 2005 installation automatically sets the ProgrammingLanguage property to its appropriate value.

© 2005 Altova GmbH

Errors

2200 The object is no longer valid.

2201 Invalid address for the return parameter was specified.

4.3.6 DatabaseConnection

See also

Properties for import and export
`File` or
`ADOConnection` or
`ODBCConnection`

Properties for import only
`DatabaseKind`
`SQLSelect`
`AsAttributes`
`ExcludeKeys`
`IncludeEmptyElements`
`NumberDateTimeFormat`

Properties for export only
`CreateMissingTables`
`CreateNew`
`TextFieldLen`

Description
`DatabaseConnection` specifies the parameters for the database connection.

Please note that the properties of the `DatabaseConnection` interface are referring to the settings of the import and export dialogs of XMLSpy 2005.

ADOConnection

See also

Property: `ADOConnection` as `String`

Description
The property `ADOConnection` contains a connection string. Either use this property or `ODBCConnection` or `File` to refer to a database.

Errors
No error codes are returned.

Example

```
Dim objSpyConn As DatabaseConnection
Set objSpyConn = objSpy.GetDatabaseSettings

Dim objADO As DataLinks
Set objADO = CreateObject("DataLinks")

If Not (objADO Is Nothing) Then
 Dim objConn As Connection
 Set objConn = objADO.PromptNew
```

```
  objSpyConn.ADOConnection = objConn.ConnectionString
End If
```

AsAttributes

See also

Property: `AsAttributes` as `Boolean`

Description
Set `AsAttributes` to true if you want to initialize all import fields to be imported as attributes. Default is false and will initialize all fields to be imported as elements. This property is used only in calls to `Application.GetDatabaseImportElementList`.

Errors
No error codes are returned.

CreateMissingTables

See also

Property: `CreateMissingTables` as `Boolean`

Description
If `CreateMissingTables` is true, tables which are not already defined in the export database will be created during export. Default is true. This property is used only when exporting to databases.

Errors
No error codes are returned.

CreateNew

See also

Property: `CreateNew` as `Boolean`

Description
Set `CreateNew` true if you want to create a new database on export. Any existing database will be overwritten. See also `DatabaseConnection.File`. Default is false. This property is used only when exporting to databases.

Errors
No error codes are returned.

DatabaseKind

See also

Property: `DatabaseKind` as `SPYDatabaseKind`

Description
Select the kind of database that gets access. The default value is `spyDB_Unspecified(7)` and is sufficient in most cases. This property is used only when importing from databases.

© 2005 Altova GmbH

Errors
No error codes are returned.

ExcludeKeys

See also

Property: `ExcludeKeys` as `Boolean`

Description
Set `ExcludeKeys` to true if you want to exclude all key columns from the import data. Default is false. This property is used only when importing from databases.

Errors
No error codes are returned.

File

See also

Property: `File` as `String`

Description
The property `File` sets the path for the database during export or import. This property can only be used in conjunction with a Microsoft Access database. Either use this property or `ODBCConnection` or `ADOConnection` to refer to the database.

See also Import and Export.

Errors
No error codes are returned.

IncludeEmptyElements

See also

Property: `IncludeEmptyElements` as `Boolean`

Description
Set `IncludeEmptyElements` to false if you want to exclude all empty elements. Default is true. This property is used only when importing from databases.

Errors
No error codes are returned.

NumberDateTimeFormat

See also

Property: `NumberDateTimeFormat` as `SPYNumberDateTimeFormat`

Description
The property `NumberDateTimeFormat` sets the format of numbers and date- and time-values. Default is `spySystemLocale`. This property is used only when importing from databases.

Errors
No error codes are returned.

ODBCConnection

See also

Property: `ODBCConnection` as `String`

Description
The property `ODBCConnection` contains a ODBC connection string. Either use this property or `ADOConnection` or `File` to refer to a database.

Errors
No error codes are returned.

SQLSelect

See also

Property: `SQLSelect` as `String`

Description
The SQL query for the import is stored in the property `SQLSelect`. This property is used only when importing from databases. See also Import and Export.

Errors
No error codes are returned.

TextFieldLen

See also

Property: `TextFieldLen` as `long`

Description
The property `TextFieldLen` sets the length for created text fields during the export. Default is 255. This property is used only when exporting to databases.

Errors
No error codes are returned.

4.3.7 Dialogs

See also

Properties and Methods

Standard automation properties
`Application`
`Parent`

Various dialog objects
`CodeGeneratorDlg`
`FileSelectionDlg`

© 2005 Altova GmbH

`SchemaDocumentationDlg`

Description
The Dialogs object provides access to different built-in dialogs of XMLSpy 2005. These dialog objects allow to initialize the fields of user dialogs before they get presented to the user or allow to simulate complete user input by your program.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

2300	The object is no longer valid.
2301	Invalid address for the return parameter was specified.

CodeGeneratorDlg

See also

Property: `CodeGeneratorDlg` as `CodeGeneratorDlg` (read-only)

Description
Get a new instance of a code generation dialog object. You will need this object to pass the necessary parameters to the code generation methods. Initial values are taken from last usage of the code generation dialog.

Errors

1111	The Dialogs object or one of its parents is no longer valid.
1100	Invalid address for the return parameter was specified.

FileSelectionDlg

See also

Property: `FileSelectionDlg` as `FileSelectionDlg` (read-only)

Description
Get a new instance of a file selection dialog object.

File selection dialog objects are passed to you with the some events that signal opening or saving of documents and projects.

Errors

1111	The Dialogs object or one of its parents is no longer valid.
1100	Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors
1111 The object is no longer valid.
1100 Invalid address for the return parameter was specified.

SchemaDocumentationDlg

See also

Property: `SchemaDocumentationDlg` as `SchemaDocumentationDlg` (read-only)

Description
Get a new instance of a dialog object that parameterizes generation of schema documentation. See Document.GenerateSchemaDocumentation for its usage.

Errors
1111 The Dialogs object or one of its parents is no longer valid.
1100 Invalid address for the return parameter was specified.

4.3.8 Document

See also

Properties and Methods

Standard automation properties
`Application`
`Parent`

Various document properties and methods
`SetActiveDocument`
`Encoding`
`SetEncoding (obsolete)`

XML validation
`IsWellFormed`
`IsValid`

Document conversion and transformation
`AssignDTD`
`AssignSchema`
`AssignXSL`
`AssignXSLFO`
`ConvertDTDOrSchema`
`GenerateDTDOrSchema`
`CreateSchemaDiagram`

© 2005 Altova GmbH

```
ExecuteXQuery
TransformXSL
TransformXSLFO
GenerateProgramCode (Enterprise Edition only)
GenerateSchemaDocumentation
```

Document export
```
GetExportElementList
ExportToText
ExportToDatabase
```

File saving and naming
```
FullName
Name
Path
GetPathName (obsolete)
SetPathName (obsolete)
Title
IsModified
Saved
SaveAs
Save
SaveInString
SaveToURL
Close
```

View access
```
CurrentViewMode
SwitchViewMode
AuthenticView
GridView
DocEditView (obsolete)
```

Access to XMLData
```
RootElement
DataRoot
CreateChild
UpdateViews
StartChanges
EndChanges
```

Description
Document objects represent XML documents opened in XMLSpy 2005.

Use one of the following properties to access documents that are already open XMLSpy 2005:
```
Application.ActiveDocument
Application.Documents
```

Use one of the following methods to open a new document in XMLSpy 2005:
```
Documents.OpenFile
Documents.OpenURL
Documents.OpenURLDialog
Documents.NewFile
Documents.NewFileFromText
SpyProjectItem.Open
Application.ImportFromDatabase
Application.ImportFromSchema
Application.ImportFromText
Application.ImportFromWord
Document.ConvertDTDOrSchema
```

```
Document.GenerateDTDOrSchema
```

Events

OnBeforeSaveDocument

See also

Event: `OnBeforeSaveDocument(objDocument as Document, objDialog as FileSelectionDlg)`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeSaveDocument(objDocument, objDialog)
End Function

' old handler - now obsolete
' return string to save to new file name
' return empty string to cancel save operation
' return nothing to save to original name
Function On_SaveDocument(objDocument, strFilePath)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeSaveDocument(objDocument, objDialog)
{
}

// old handler - now obsolete
// return string to save to new file name
// return empty string to cancel save operation
// return nothing to save to original name
function On_SaveDocument(objDocument, strFilePath)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (27, ...)  // nEventId = 27
```

Description

This event gets fired on any attempt to save a document. The file selection dialog object is initialized with the name chosen for the document file. You can modify this selection. To continue saving the document leave the `FileSelectionDlg.DialogAction` property of *io_objDialog* at its default value `spyDialogOK`. To abort saving of the document set this property to `spyDialogCancel`.

OnBeforeCloseDocument

See also

Event: `OnBeforeCloseDocument(objDocument as Document)as Boolean`

XMLSpy scripting environment - VBScript:

```
Function On_BeforeCloseDocument(objDocument)
     ' On_BeforeCloseDocument = False  ' to prohibit closing of
document
```


© 2005 Altova GmbH

```
End Function
```

XMLSpy scripting environment - JScript:

```
function On_BeforeCloseDocument(objDocument)
{
    // return false;  /* to prohibit closing of document */
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (28, ...)  // nEventId = 28
```

Description
This event gets fired on any attempt to close a document. To prevent the document from being closed return false.

OnCloseDocument

See also

Event: `OnCloseDocument(objDocument as Document)`

XMLSpy scripting environment - VBScript:

```
Function On_Close Document(objDocument)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_Close Document(objDocument)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (8, ...)   // nEventId = 8
```

Description
This event gets fired as a result of closing a document. Do not modify the document from within this event.

OnViewActivation

See also

Event: `OnViewActivation(objDocument as Document, eViewMode as SPYViewModes, bActivated as Boolean)`

XMLSpy scripting environment - VBScript:

```
Function On_ViewActivation(objDocument, eViewMode, bActivated)
End Function
```

XMLSpy scripting environment - JScript:

```
function On_ViewActivation(objDocument, eViewMode, bActivated)
{
}
```

XMLSpy IDE Plugin:

```
IXMLSpyPlugIn.OnEvent (29, ...)  // nEventId = 29
```

Description
This event gets fired whenever a view of a document becomes visible (i.e. becomes the active view) or invisible (i.e. another view becomes the active view or the document gets closed). However, the first view activation event after a document gets opened cannot be received, since there is no document object to get the event from. Use the `Application.OnDocumentOpened` event instead.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

1400	The object is no longer valid.
1407	Invalid address for the return parameter was specified.

AssignDTD

See also

Method: `AssignDTD(strDTDFile` as String, `bDialog` as Boolean`)`

Description
The method places a reference to the DTD file "strDTDFile" into the document. Note that no error occurs if the file does not exist, or is not accessible. If `bDialog` is true XMLSpy 2005 presents a dialog to set the file.

See also Simple document access.

Errors

1400	The object is no longer valid.
1409	You are not allowed to assign a DTD to the document.

AssignSchema

See also

Method: `AssignSchema` (`strSchemaFile` as String, `bDialog` as Boolean)

Description
The method places a reference to the schema file "strSchemaFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If `bDialog` is true XMLSpy 2005 presents a dialog to set the file.

See also Simple document access.

Errors

© 2005 Altova GmbH

1400 The object is no longer valid.

1409 You are not allowed to assign a schema file to the document.

AssignXSL

See also

Method: `AssignXSL` (*`strXSLFile`* as String, *`bDialog`* as Boolean)

Description
The method places a reference to the XSL file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If `bDialog` is true XMLSpy 2005 presents a dialog to set the file.

Errors

1400 The object is no longer valid.

1409 You are not allowed to assign an XSL file to the document.

AssignXSLFO

See also

Method: `AssignXSLFO` (*`strXSLFOFile`* as String, *`bDialog`* as Boolean)

Description
The method places a reference to the XSLFO file "strXSLFile" into the document. Note that no error occurs if the file does not exist or is not accessible. If `bDialog` is true XMLSpy 2005 presents a dialog to set the file.

Errors

1400 The object is no longer valid.

1409 You are not allowed to assign an XSL file to the document.

AuthenticView

See also

Method: `AuthenticView` as `AuthenticView` (read-only)

Description
Returns an object that gives access to properties and methods specific to Authentic view. The object returned is only valid if the current document is opened in Authentic view mode. The lifetime of an object ends with the next view switch. Any attempt to access objects or any of its children afterwards will result in an error indicating that the object is invalid.

`AuthenticView` and `DocEditView` both provide automation access to the Authentic view mode of XMLSpy. Functional overlap is intentional. A future version of Authentic View will include all functionality of `DocEditView` and its sub-objects, thereby making usage of `DocEditView` obsolete.

Errors

1400 The object is no longer valid.

1417 Document needs to be open in authentic view mode.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' secure access to authentic view object
' ---------------------------------------
Dim objDocument
Set objDocument = Application.ActiveDocument
If (Not objDocument Is Nothing) Then
	' we have an active document, now check for view mode
	If (objDocument.CurrentViewMode <> spyViewContent) Then
		If (Not objDocument.SwitchViewMode (spyViewContent)) Then
			MsgBox "Active document does not support authentic view
mode"
		Else
			' now it is safe to access the authentic view object
			Dim objAuthenticView
			Set objAuthenticView = objDocument.AuthenticView
			' now use the authentic view object

		End If
	End If
Else
	MsgBox "No document is open"
End If
```

Close

See also

Method: `Close` (*`bDiscardChanges`* as Boolean)

Description

To close the document call this method. If `bDiscardChanges` is true and the document is modified, the document will be closed but not saved.

Errors

1400 The object is no longer valid.

1401 Document needs to be saved first.

ConvertDTDOrSchema

See also

Method: `ConvertDTDOrSchema` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nFrequentElements`* as `SPYFrequentElements`)

Parameters

`nFormat`
Sets the schema output format to DTD, DCD, XMLData, BizTalk or W3C.

`nFrequentElements`
Create complex elements as elements or complex types.

Description

 © 2005 Altova GmbH

`ConvertDTDOrSchema` takes an existing schema format and converts it into a different format.

Errors

1400 The object is no longer valid.

1412 Error during conversion.

CreateChild

See also

Method: `CreateChild` (*`nKind`* as `SPYXMLDataKind`) as `XMLData`

Return Value
The method returns the new `XMLData` object.

Description
To create a new `XMLData` object use the `CreateChild()` method. See also Using XMLData.

Errors

1400 The object is no longer valid.

1404 Cannot create XMLData object.

1407 Invalid address for the return parameter was specified.

CreateSchemaDiagram

See also

Method: `CreateSchemaDiagram` (*`nKind`* as `SPYSchemaDefKind`, *`strName`* as String, *`strFile`* as String)

Return Value
None.

Description
The method creates a diagram of the schema type `strName` of kind `nKind` and saves the output file into `strFile`.

Errors

1400 The object is no longer valid.

1414 Failed to save diagram.

1415 Invalid schema definition type specified.

CurrentViewMode

See also

Method: `CurrentViewMode` as `SPYViewModes`

Description
The property holds the current view mode of the document. See also `Document.SwitchViewMode`.

Errors

1400 The object is no longer valid.

1407 Invalid address for the return parameter was specified.

DataRoot

See also

Property: `DataRoot` as `XMLData` (read-only)

Description
This property provides access to the document's first XMLData object of type *spyXMLDataElement*. This is typically the root element for all document content data. See XMLSpyDocument.RootElement to get the root element of the whole document including XML prolog data.

Errors

1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.

DocEditView

See also

Method: `DocEditView` as `DocEditView`

Description
Holds a reference to the current Authentic object.

Errors

1400 The object is no longer valid.

1407 Invalid address for the return parameter was specified.

1417 Document needs to be open in authentic view mode.

Encoding

See also

Property: `Encoding` as String

Description
This property provides access to the document's encoding value. However, this property can only be accessed when the document is opened in *spyViewGrid*, *spyViewText* or *spyViewContent*. See CurrentViewMode on how to detect that a document's actual view mode.

This property makes the method SetEncoding obsolete.

Possible values are, for example:

8859-1,
8859-2,
ASCII, ISO-646,

© 2005 Altova GmbH

850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

Errors

1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.

1416 Operation not supported in current view mode.

EndChanges

See also

Method: `EndChanges()`

Description

Use the method `EndChanges` to display all changes since the call to `Document.StartChanges`.

Errors

1400 The object is no longer valid.

ExportToDatabase

See also

Method: `ExportToDatabase` (*`pFromChild`* as `XMLData`, *`pExportSettings`* as `ExportSettings`, *`pDatabase`* as `DatabaseConnection)`

Description

`ExportToDatabase` exports the XML document starting with the element `pFromChild`. The parameter `pExportSettings` defines the behaviour of the export (see `Application.GetExportSettings`). The parameter `pDatabase` specifies the destination of the export (see `Application.GetDatabaseSettings`).

Errors

1400 The object is no longer valid.

1407 Invalid parameter or invalid address for the return parameter was specified.

1416 Error during export.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

'set the behaviour of the export with ExportSettings
```

```
Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings

'set the destination with DatabaseConnection
Dim objDB As DatabaseConnection
Set objDB = objSpy.GetDatabaseSettings

objDB.CreateMissingTables = True
objDB.CreateNew = True
objDB.File = "C:\Export.mdb"

objDoc.ExportToDatabase objDoc.RootElement, objExpSettings, objDB
If Err.Number <> 0 Then
 a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
   "Description: " & Err.Description)
End If
```

ExportToText

See also

Method: `ExportToText` (*`pFromChild`* as `XMLData`, *`pExportSettings`* as `ExportSettings`, *`pTextSettings`* as `TextImportExportSettings)`

Description

`ExportToText` exports tabular information from the document starting at `pFromChild` into one or many text files. Columns of the resulting tables are generated in alphabetical order of the column header names. Use `GetExportElementList` to learn about the data that will be exported. The parameter `pExportSettings` defines the specifics for the export. Set the property `ExportSettings.ElementList` to the - possibly modified - list returned by `GetExportElementList` to avoid exporting all contained tables. The parameter `pTextSettings` defines the options specific to text export and import. You need to set the property `TextImportExportSettings.DestinationFolder` before you call `ExportToText`.

See also `Import and export of data`.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.
1416	Error during export.

Example

```
' ----------------------------------------------------------
' VBA client code fragment - export document to text files
' ----------------------------------------------------------
Dim objDoc As Document
Set objDoc = objSpy.ActiveDocument

Dim objExpSettings As ExportSettings
Set objExpSettings = objSpy.GetExportSettings
objExpSettings.ElementList = objDoc.GetExportElementList(
       objDoc.RootElement,
       objExpSettings)

Dim objTextExp As TextImportExportSettings
Set objTextExp = objSpy.GetTextExportSettings
objTextExp.HeaderRow = True
```

 © 2005 Altova GmbH

```
 objTextExp.DestinationFolder = "C:\Exports"

 On Error Resume Next
 objDoc.ExportToText objDoc.RootElement, objExpSettings, objTextExp

 If Err.Number <> 0 Then
  a = MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
"Description: "     & Err.Description)
 End If
```

ExecuteXQuery

See also

Method: `ExecuteXQuery` (*`strXMLFileName`* as String)

Description
Execute the XQuery statements contained in the document. Use the XML file specified as XML source for the transformation. If your XQuery script does not use an XML source set the parameter `strXMLFileName` to an empty string.

Errors

1400	The document object is no longer valid.
1423	XQuery transformation error.
1424	Not all files required for operation could be loaded. Most likely, the file specified in `strXMLFileName` does not exist or is not valid.

FullName

See also

Property: `FullName` as String

Description
This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName and SetPathName obsolete.

Errors

1400	The document object is no longer valid.
1402	Empty string has been specified as full file name.

GenerateDTDOrSchema

See also

Method: `GenerateDTDOrSchema` (*`nFormat`* as `SPYDTDSchemaFormat`, *`nValuesList`* as integer, *`nDetection`* as `SPYTypeDetection`, *`nFrequentElements`* as `SPYFrequentElements`)

Parameters
`nFormat`
Sets the schema output format to DTD, DCD, XMLData, BizTalk or W3C.

`nValuesList`
Set to 0 (zero) for unlimited.

`nDetection`
Specifies attribute/element detection.

`nFrequentElements`
Create complex elements as elements or complex types.

Description
To generate a DTD or schema from the current document use the `GenerateDTDOrSchema` method.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

GenerateProgramCode

Method: `GenerateProgramCode` (*`objDlg`* as `CodeGeneratorDlg`)

Description
Generate Java, C++ or C# class files from the XML Schema definitions in your document. A `CodeGeneratorDlg` object is used to pass information to the code generator. The generation process can be configured to allow user interaction or run without further user input.

Errors

1400	The document object is no longer valid.
1407	An empty file name has been specified.
1421	Feature not available in this edition

GenerateSchemaDocumentation

Method: `GenerateSchemaDocumentation` (*`objDlg`* as `SchemaDocumentationDlg`)

Description
Generate documentation for a schema definition file in MS-Word or HTML format. The parameter objDlg is used to parameterize the generation process. Use `Dialogs.SchemaDocumentationDlg` to get an initialized set of options. As a minimum, you will need to set the property `SchemaDocumentationDlg.OutputFile` before starting the generation process.

Errors

1400	The document object is no longer valid.
1407	Invalid parameters have been passed or an empty file name has been specified as output target.
1417	The document is not opened in schema view, maybe it is not an '.xsd' file.
1421	Feature is not available in this edition.
1422	Error during generation

© 2005 Altova GmbH

GetExportElementList

See also

Method: GetExportElementList (*pFromChild* as XMLData, *pExportSettings* as ExportSettings) as ElementList

Description
GetExportElementList creates a collection of elements to export from the document, depending on the settings in pExportSettings and starting from the element pFromChild. The function returns a collection of ElementListItems where the properties ElementListItem.Name contain the names of the tables that can be exported from the document. The property ElementListItem.FieldCount contains the number of columns in the table. The property ElementListItem.RecordCount contains the number of records in the table. The property ElementListItem.ElementKind is unused.

See also Import and export of data.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

GetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

See also

Method: GetPathName() as String

Description
The method GetPathName gets the path of the active document.

See also Document.SetPathName (obsolete).

GridView

See also

Property: GridView as GridView

Description
This property provides access to the grid view functionality of the document.

Errors

1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.
1417 Document needs to be open in enhanced grid view mode.

IsModified

See also

Property: `IsModified` as Boolean

Description
True if the document is modified.

Errors
1400 The object is no longer valid.
1407 Invalid address for the return parameter was specified.

IsValid

See also

Method: `IsValid` (*`strError`* as Variant, *`nErrorPos`* as Variant, *`pBadData`* as Variant) as Boolean

Return Value
True if the document is valid, false if not.

Description
`IsValid()` validates the document against its associated schema or DTD. `strError` and `nErrorPos` give you the same information as XMLSpy 2005 if you validate the file within the editor.

Errors
1400 The object is no longer valid.
1407 Invalid parameter or invalid address for the return parameter was specified.
1408 Unable to validate file.

Examples

For an example written in VBA see Overview - Simple document access

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' ---------------------------------------
Dim errorText
Dim errorPos
Dim badData

' validate the active document
Set doc = Application.ActiveDocument
valid = doc.IsValid(errorText, errorPos, badData)
```

 © 2005 Altova GmbH

```
If (Not valid) Then
	' create the validation error message text
	Text = errorText

	If ( Not IsEmpty(badData) ) Then
		Text = Text & "(" & badData.Name & "/" & badData.TextValue & ")"
	End If

	Call MsgBox("validation error[" & errorPos & "]: " & Text)
End If
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// ---------------------------------------
// define as arrays to support their usage as return parameters
var errorText = new Array(1);
var errorPos = new Array(1);
var badData = new Array(1);

// validate the active document
var doc = Application.ActiveDocument;
var valid = doc.IsValid(errorText, errorPos, badData);

if (! valid)
{
	// compose the error description
	var text = errorText;

	// access that XMLData object only if filled in
	if (badData[0] != null)
		text += "(" + badData[0].Name + "/" + badData[0].TextValue + ")";

	MsgBox("validation error[" + errorPos + "]: " + text);
}
```

IsWellFormed

See also

Method: `IsWellFormed` (*`pData`* as `XMLData`, *`bWithChildren`* as `Boolean`, *`strError`* as Variant, *`nErrorPos`* as Variant, *`pBadXMLData`* as Variant) as Boolean

Return Value

True if the document is well formed.

Description

`IsWellFormed` checks the document for well-formedness starting at the element `pData`.

If the document is not well formed, `strError` contains an error message, `nErrorPos` the position in the file and `pBadXMLData` holds a reference to the element which breaks the well-formedness. These out-parameters are defined as VARIANTs to support scripting languages like VBScript.

Errors

1400	The object is no longer valid.
1407	Invalid parameter or invalid address for the return parameter was specified.

Example

See IsValid.

Name

See also

Property: `Name` as String (read-only)

Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Documents` (read-only)

Description
Access the parent of the document object.

Errors
2400 The document object is no longer valid.
2401 Invalid address for the return parameter was specified.

Path

See also

Property: `Path` as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

RootElement

See also

Property: `RootElement` as `XMLData` (read-only)

Description

© 2005 Altova GmbH

The property `RootElement` provides access to the root element of the XML structure of the document including the XML prolog data. To access the first element of a document's content navigate to the first child of kind *spyXMLDataElement* or use the Document.DataRoot property.

Errors

1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.

Save

See also

Method: `Save()`

Description

The method writes any modifications of the document to the associated file. See also `Document.FullName`.

Errors

1400 The document object is no longer valid.

1407 An empty file name has been specified.

1403 Error when saving file, probably the file name is invalid.

SaveAs

See also

Method: `SaveAs` (*`strFileName`* as String)

Description

Save the document to the file specified. If saving was successful, the FullName property gets set to the specified file name.

Errors

1400 The document object is no longer valid.

1407 An empty file name has been specified.

1403 Error when saving file, probably the file name is invalid.

Saved

See also

Property: `Saved` as Boolean (read-only)

Description

This property can be used to check if the document has been saved after the last modifications. It returns the negation of IsModified.

Errors

1400 The document object is no longer valid.

1407 Invalid address for the return parameter was specified.

SaveInString

See also

Method: `SaveInString` (*`pData`* as `XMLData`, *`bMarked`* as Boolean) as String

Parameters
`pData`
`XMLData` element to start. Set `pData` to `Document.RootElement` if you want to copy the complete file.

`bMarked`
If `bMarked` is true, only the elements selected in the grid view are copied.

Return Value
Returns a string with the XML data.

Description
`SaveInString` starts at the element `pData` and converts the `XMLData` objects to a string representation.

Errors

1400 The object is no longer valid.

1407 Invalid parameter or invalid address for the return parameter was specified.

SaveToURL

See also

Method: `SaveToURL` (*`strURL`* as String, *`strUser`* as String, *`strPassword`* as String)

Return Value

Description
`SaveToURL()` writes the document to the URL `strURL`. This method does not set the permanent file path of the document.

Errors

1400 The object is no longer valid.

1402 Invalid URL specified.

1403 Error while saving to URL.

SetActiveDocument

See also

Method: `SetActiveDocument()`

 © 2005 Altova GmbH

Description
The method sets the document as the active and brings it to the front.

Errors
1400 The object is no longer valid.

SetEncoding (obsolete)

Superseded by Document.Encoding

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetEncoding("UTF-16");
// use now:
Application.ActiveDocument.Encoding = "UTF-16";
```

See also

Method: `SetEncoding` (*`strEncoding`* as String)

Description
`SetEncoding` sets the encoding of the document like the menu item "File/Encoding..." in XMLSpy 2005. Possible values for `strEncoding` are, for example:

8859-1,
8859-2,
ASCII, ISO-646,
850,
1252,
1255,
SHIFT-JIS, MS-KANJI,
BIG5, FIVE,
UTF-7,
UTF-8,
UTF-16

SetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

See also

Method: `SetPathName` (*`strPath`* as String)

Description
The method `SetPathName` sets the path of the active document. `SetPathName` only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

StartChanges

See also

Method: `StartChanges()`

Description
`After StartChanges` is executed XMLSpy 2005 will not update its editor windows until `Document.EndChanges` is called. This increases performance of complex tasks to the XML structure.

Errors

1400 The object is no longer valid.

SwitchViewMode

See also

Method: `SwitchViewMode` (*`nMode`* as `SPYViewModes`) as Boolean

Return value
Returns true if view mode is switched.

Description
The method sets the current view mode of the document in XMLSpy 2005. See also `Document.CurrentViewMode`.

Errors

1400 The object is no longer valid.

1407 Invalid address for the return parameter was specified.

1417 Invalid view mode specified.

© 2005 Altova GmbH

Title

See also

Property: `Title` as String (read-only)

Description
`Title` contains the file name of the document. To get the path and filename of the file use `FullName`.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

TransformXSL

See also

Method: `TransformXSL()`

Description
`TransformXSL` processes the XML document via the associated XSL file. See `Document.AssignXSL` on how to place a reference to a XSL file into the document.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

TransformXSLFO

See also

Method: `TransformXSLFO()`

Description
`TransformXSLFO` processes the XML document via the associated XSLFO file. See `AssignXSLFO` on how to place a reference to a XSLFO file into the document. You need to assign a FOP processor to XMLSpy 2005 before you can use this method.

Errors
1400 The document object is no longer valid.
1411 Error during transformation process.

UpdateViews

See also

Method: `UpdateViews()`

Description
To redraw the Enhanced Grid View and the Tree View call `UpdateViews`. This can be

important after you changed the XMLData structure of a document. This method does not redraw the text view of XMLSpy 2005.

Errors
1400 The document object is no longer valid.

4.3.9 Documents

See also

Properties
Count
Item

Methods
NewFile
OpenFile
OpenURL
OpenURLDialog
NewFileFromText

Description
This object represents the set of documents currently open in XMLSpy 2005. Use this object to open further documents or iterate through already opened documents.

Examples

```
' ---------------------------------------
' XMLSpy scripting environment - VBScript
' iterate through open documents
' ---------------------------------------
Dim objDocuments
Set objDocuments = Application.Documents

For Each objDoc In objDocuments
     'do something useful with your document
     objDoc.SetActiveDocument()
Next
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// close all open documents
// ---------------------------------------
for (var iter = new Enumerator (Application.Documents);
     ! iter.atEnd();
     iter.moveNext())
{
     // MsgBox ("Closing file " + iter.item().Name);
     iter.item().Close (true);
}
```

Count

See also

Property: Count as long

© 2005 Altova GmbH

Description
Count of open documents.

Item

See also

Method: `Item` (*n* as long) as `Document`

Description
Gets the document with the index n in this collection. Index is 1-based.

NewFile

See also

Method: `NewFile` (*`strFile`* as String, *`strType`* as String) as `Document`

Parameters
`strFile`
Full path of new file.

`strType`
Type of new file as string (i.e. "xml", "xsd", ...)

Return Value
Returns the new file.

Description
`NewFile` creates a new file of type `strType` (i.e. "xml"). The newly created file is also the ActiveDocument.

NewFileFromText

See also

Method: `NewFileFromText` (*`strText`* as String, *`strType`* as String) as `Document`

Parameters
`strText`
The content of the new document in plain text.

`strType`
Type of the document to create (i.e. "xml").

Return Value
The method returns the new document.

Description
NewFileFromText creates a new document with strText as its content.

OpenFile

See also

Method: `OpenFile` (*`strPath`* as String, *`bDialog`* as Boolean) as `Document`

Parameters
`strPath`
Path and file name of file to open.

`bDialog`
Show dialogs for user input.

Return Value
Returns the opened file on success.

Description
`OpenFile` opens the file `strPath`. If `bDialog` is TRUE, a file-dialog will be displayed.

Example

```
Dim objDoc As Document
Set objDoc = objSpy.Documents.OpenFile(strFile, False)
```

OpenURL

See also

Method: `OpenURL` (*`strURL`* as String, *`nURLType`* as `SPYURLTypes`, *`nLoading`* as `SPYLoading`, *`strUser`* as String, *`strPassword`* as String) as `Document`

Parameters
`strURL`
URL to open as document.

`nURLType`
Type of document to open. Set to -1 for auto detection.

`nLoading`
Set `nLoading` to 0 (zero) if you want to load it from cache or proxy. Otherwise set `nLoading` to 1.

`strUser`
Name of the user if required. Can be empty.

`strPassword`
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
`OpenURL` opens the URL strURL.

OpenURLDialog

See also

Method: `OpenURLDialog` (*`strURL`* as String, *`nURLType`* as `SPYURLTypes`, *`nLoading`* as `SPYLoading`, *`strUser`* as String, *`strPassword`* as String) as `Document`

© 2005 Altova GmbH

Parameters
`strURL`
URL to open as document.

`nURLType`
Type of document to open. Set to -1 for auto detection.

`nLoading`
Set `nLoading` to 0 (zero) if you want to load it from cache or proxy. Otherwise set `nLoading` to 1.

`strUser`
Name of the user if required. Can be empty.

`strPassword`
Password for authentification. Can be empty.

Return Value
The method returns the opened document.

Description
`OpenURLDialog` displays the "open URL" dialog to the user and presets the input fields with the given parameters.

4.3.10 ElementList

See also

Properties
`Count`
`Item`

Methods
`RemoveElement`

Description
Element lists are used for different purposes during export and import of data. Depending on this purpose, different properties of `ElementListItem` are used.

It can hold
- a list of table names returned by a call to `Application.GetDatabaseTables`,
- a list of field names retuned by a call to `Application.GetDatabaseImportElementList` or `Application.GetTextImportElementList`,
- a field name filter list used in `Application.ImportFromDatabase` and `Application.ImportFromText`,
- a list of table names and counts for their rows and columns as returned by calls to `GetExportElementList` or
- a field name filter list used in `Document.ExportToDatabase` and `Document.ExportToText`.

Count

See also

Property: `Count` as long (read-only)

Description
Count of elements in this collection.

Item

See also

Method: `Item(n as long) as ElementListItem`

Description
Gets the element with the index n from this collection. The first item has index 1.

RemoveElement

See also

Method: `RemoveElement(Index as long)`

Description
`RemoveElement` removes the element Index from the collection. The first Item has index 1.

4.3.11 ElementListItem

See also

Properties
`Name`

`ElementKind`

`FieldCount`
`RecordCount`

Description
An element in an `ElementList`. Usage of its properties depends on the purpose of the element list. For details see `ElementList`.

ElementKind

See also

Property: `ElementKind` as `SPYXMLDataKind`

Description
Specifies if a field should be imported as XML element (data value of `spyXMLDataElement`) or attribute (data value of `spyXMLDataAttr`).

FieldCount

See also

Property: `FieldCount` as long (read-only)

Description

© 2005 Altova GmbH

Count of fields (i.e. columns) in the table described by this element. This property is only valid after a call to `Document.GetExportElementList`.

Name

See also

Property: `Name` as String (read-only)

Description
Name of the element. This is either the name of a table or a field, depending on the purpose of th element list.

RecordCount

See also

Property: `RecordCount` as long (read-only)

Description
Count of records (i.e. rows) in the table described by this element. This property is only valid after a call to `Document.GetExportElementList`.

4.3.12 ExportSettings

See also

Properties

`ElementList`

`EntitiesToText`

`ExportAllElements`
`SubLevelLimit`

`FromAttributes`
`FromSingleSubElements`
`FromTextValues`

`CreateKeys`
`IndependentPrimaryKey`

`Namespace`

Description
`ExportSettings` contains options used during export of XML data to a database or text file. See Import and export of data for a general overview.

CreateKeys

See also

Property: `CreateKeys` as Boolean

Description
This property turns creation of keys (i.e. primary key and foreign key) on or off. Default is True.

ElementList

See also

Property: `ElementList` as `ElementList`

Description
Default is empty list. This list of elements defines which fields will be exported. To get the list of available fields use `Document.GetExportElementList`. It is possible to prevent exporting columns by removing elements from this list with `ElementList.RemoveElement` before passing it to `Document.ExportToDatabase` or `Document.ExportToText`.

EntitiesToText

See also

Property: `EntitiesToText` as Boolean

Description
Defines if XML entities should be converted to text or left as they are during export. Default is True.

ExportAllElements

See also

Property: `ExportAllElements` as Boolean

Description
If set to `true`, all elements in the document will be exported. If set to `false`, then `ExportSettings.SubLevelLimit` is used to restrict the number of sub levels to export. Default is `true`.

FromAttributes

See also

Property: `FromAttributes` as Boolean

Description
`Set FromAttributes` to false if no export data should be created from attributes. Default is True.

FromSingleSubElements

See also

Property: `FromSingleSubElements` as Boolean

Description
`Set FromSingleSubElements` to false if no export data should be created from elements.

© 2005 Altova GmbH

Default is True.

FromTextValues

See also

Property: `FromTextValues` as Boolean

Description
`Set FromTextValues` to false if no export data should be created from text values. Default is True.

IndependentPrimaryKey

See also

Property: `IndependentPrimaryKey` as Boolean

Description
Turns creation of independent primary key counter for every element on or off. If `ExportSettings.CreateKeys` is False, this property will be ignored. Default is True.

Namespace

See also

Property: `Namespace` as `SPYExportNamespace`

Description
The default setting removes all namespace prefixes from the element names. In some database formats the colon is not a legal character. Default is `spyNoNamespace`.

SubLevelLimit

See also

Property: `SubLevelLimit` as Integer

Description
Defines the number of sub levels to include for the export. Default is 0. This property is ignored if `ExportSettings.ExportAllElements` is `true`.

4.3.13 FileSelectionDlg

See also

Properties and Methods

Standard automation properties
`Application`
`Parent`

Dialog properties
`FullName`

Acceptance or cancellation of action that caused event
`DialogAction`

Description
The dialog object allows you to receive information about an event and pass back information to the event handler in the same way as with a user dialog. Use the `FileSelectionDlg.FullName` to select or modify the file path and set the `FileSelectionDlg.DialogAction` property to cancel or agree with the action that caused the event.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

DialogAction

Property: `DialogAction` as `SPYDialogAction`

Description
If you want your script to perform the file selection operation without any user interaction necessary, simulate user interaction by either setting the property to *spyDialogOK(0)* or *spyDialogCancel(1)*.
To allow your script to fill in the default values but let the user see and react on the dialog, use the value *spyDialogUserInput(2)*. If you receive a FileSelectionDlg object in an event handler, *spyDialogUserInput(2)* is not supported and will be interpreted as *spyDialogOK(0)*.

Errors
2400 The object is no longer valid.
2401 Invalid value for dialog action or invalid address for the return parameter was specified.

FullName

Property: `FullName` as String

Description
Access the full path of the file the gets selected by the dialog. Most events that pass a FileSelectionDlg object to you allow you modify this value and thus influence the action that caused the event (e.g. load or save to a different location).

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

© 2005 Altova GmbH

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors
2400 The object is no longer valid.
2401 Invalid address for the return parameter was specified.

4.3.14 GridView

See also

Methods
`Deselect`
`Select`

`SetFocus`

Properties
`CurrentFocus`

`IsVisible`

Description
`GridView` Class

Events

OnBeforeDrag

See also

Event: `OnBeforeDrag() as Boolean`

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrag()
     ' On_BeforeStartEditing = False  ' to prohibit dragging
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrag()
{
     // return false;  /* to prohibit dragging */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (4, ...)   // nEventId = 4
```

Description

This event gets fired on an attempt to drag an XMLData element on the grid view. Return *false* to prevent dragging the data element to a different position.

OnBeforeDrop

See also

Event: `OnBeforeDrop(objXMLData as XMLData) as Boolean`

XMLSpy scripting environment - VBScript:
```
Function On_BeforeDrop(objXMLData)
     ' On_BeforeStartEditing = False  ' to prohibit dropping
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeDrop(objXMLData)
{
     // return false;  /* to prohibit dropping */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (5, ...)   // nEventId = 5
```

Description
This event gets fired on an attempt to drop a previously dragged XMLData element on the grid view. Return *false* to prevent the data element to be moved from its original position to the drop destination position.

OnBeforeStartEditing

See also

Event: `OnBeforeStartEditing(objXMLData as XMLData, bEditingName as Boolean)as Boolean`

XMLSpy scripting environment - VBScript:
```
Function On_BeforeStartEditing(objXMLData, bEditingName)
     ' On_BeforeStartEditing = False  ' to prohibit editing the field
End Function
```

XMLSpy scripting environment - JScript:
```
function On_BeforeStartEditing(objXMLData, bEditingName)
{
     // return false;  /* to prohibit editing the field */
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (1, ...)   // nEventId = 1
```

Description
This event gets fired before the editing mode for a grid cell gets entered. If the parameter *bEditingName* is true, the name part of the element will be edited, it its value is false, the value part will be edited.

 © 2005 Altova GmbH

OnEditingFinished

See also

Event: `OnEditingFinished(objXMLData as XMLData, bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:
```
Function On_EditingFinished(objXMLData, bEditingName)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_EditingFinished(objXMLData, bEditingName)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (2, ...)   // nEventId = 2
```

Description
This event gets fired when the editing mode of a grid cell gets left. The parameter *bEditingName* specifies if the name part of the element has been edited.

OnFocusChanged

See also

Event: `OnFocusChanged(objXMLData as XMLData, bSetFocus as Boolean, bEditingName as Boolean)`

XMLSpy scripting environment - VBScript:
```
Function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
End Function
```

XMLSpy scripting environment - JScript:
```
function On_FocusChanged(objXMLData, bSetFocus, bEditingName)
{
}
```

XMLSpy IDE Plugin:
```
IXMLSpyPlugIn.OnEvent (3, ...)   // nEventId = 3
```

Description
This event gets fired whenever a grid cell receives or looses the cursor focus. If the parameter *bEditingName* is *true*, focus of the name part of the grid element has changed. Otherwise, focus of the value part has changed.

CurrentFocus

See also

Property: `CurrentFocus as XMLData`

Description

Holds the XML element with the current focus. This property is read-only.

Deselect

See also

Method: `Deselect(pData as XMLData)`

Description
Deselects the element `pData` in the grid view.

IsVisible

See also

Property: `IsVisible` as Boolean

Description
True if the grid view is the active view of the document. This property is read-only.

Select

See also

Method: `Select (pData as XMLData)`

Description
Selects the XML element `pData` in the grid view.

SetFocus

See also

Method: `SetFocus (pFocusData as XMLData)`

Description
Sets the focus to the element `pFocusData` in the grid view.

4.3.15 SchemaDocumentationDlg

See also

Properties and Methods

Standard automation properties
`Application`
`Parent`

Interaction and visibility properties
`OutputFile`
`OutputFileDialogAction`
`OptionsDialogAction`
`ShowProgressBar`
`ShowResult`

© 2005 Altova GmbH

Document generation options and methods
`OutputFormat`

`IncludeEverything`
`IncludeIndex`
`IncludeGlobalElements`
`IncludeLocalElements`
`IncludeGroups`
`IncludeComplexTypes`
`IncludeSimpleTypes`
`IncludeAttributeGroups`
`IncludeRedefines`

`AllDetails`
`ShowDiagram`
`ShowNamespace`
`ShowType`
`ShowChildren`
`ShowUsedBy`
`ShowProperties`
`ShowSingleFacets`
`ShowPatterns`
`ShowEnumerations`
`ShowAttributes`
`ShowIdentityConstraints`
`ShowAnnotations`
`ShowSourceCode`

Description
This object combines all options for schema document generation as they are available through user interface dialog boxes in XMLSpy 2005. The document generation options are initialized with the values used during the last generation of schema documentation. However, before using the object you have to set the `OutputFile` property to a valid file path. Use `OptionsDialogAction`, `OutputFileDialogAction` and `ShowProgressBar` to specify the level of user interaction desired. You can use `IncludeEverything` and `AllDetails` to set whole option groups at once or the individual properties to operate on a finer granularity.

Application

See also

Property: `Application` as `Application` (read-only)

Description
Access the XMLSpy 2005 application object.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeEverything

See also

Method: `IncludeEverything` (`i_bInclude` as `Boolean`)

Description
Use this method to mark or unmark all include options.

Errors
2900 The object is no longer valid.

AllDetails

See also

Method: `AllDetails` (`i_bDetailsOn` as `Boolean`)

Description
Use this method to turn all details options on or off.

Errors
2900 The object is no longer valid.

IncludeAttributeGroups

See also

Property: `IncludeAttributeGroups` as `Boolean`

Description
Set this property to `true`, to include attribute groups into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

IncludeComplexTypes

See also

Property: `IncludeComplexTypes` as `Boolean`

Description
Set this property to `true`, to include complex types into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

© 2005 Altova GmbH

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeGlobalElements

See also

Property: `IncludeGlobalElements` as `Boolean`

Description

Set this property to `true`, to include global elements into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeGroups

See also

Property: `IncludeGroups` as `Boolean`

Description

Set this property to `true`, to include groups into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeIndex

See also

Property: `IncludeIndex` as `Boolean`

Description

Set this property to `true`, to include an index into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeLocalElements

See also

Property: `IncludeLocalElements` as `Boolean`

Description
Set this property to `true`, to include local elements into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeRedefines

See also

Property: `IncludeRedefines` as `Boolean`

Description
Set this property to `true`, to include redefines into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

IncludeSimpleTypes

See also

Property: `IncludeSimpleTypes` as `Boolean`

Description
Set this property to `true`, to include simple types into the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

OptionsDialogAction

See also

Property: `OptionsDialogAction` as `SPYDialogAction`

© 2005 Altova GmbH

Description
To allow your script to fill in the default values and let the user see and react on the dialog, set this property to the value *spyDialogUserInput(2)*. If you want your script to define all the options in the schema documentation dialog without any user interaction necessary, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

2900	The object is no longer valid.
2901	Invalid value has been used to set the property. Invalid address for the return parameter was specified.

OutputFile

See also

Property: `OutputFile` as `String`

Description
Full path and name of the file that will contain the generated documentation. In case of HTML output, additional '.png' files will be generated based on this filename. The default value for this property is an empty string and needs to be replaced before using this object in a call to `Document.GenerateProgramCode`.

Errors

2900	The object is no longer valid.
2901	Invalid address for the return parameter was specified.

OutputFileDialogAction

See also

Property: `OutputFileDialogAction` as `SPYDialogAction`

Description
To allow the user to select the output file with a file selection dialog, set this property to *spyDialogUserInput(2)*. If the value stored in OutputFile should be taken and no user interaction should occur, use *spyDialogOK(0)*. Default is *spyDialogOK.*

Errors

2900	The object is no longer valid.
2901	Invalid value has been used to set the property. Invalid address for the return parameter was specified.

OutputFormat

See also

Property: `OutputFormat` as `SPYSchemaDocumentationFormat`

Description
Defines the kind of documentation that will be generated, either HTML style or MS-Word. The property gets initialized with the value used during the last call to

`Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid value has been used to set the property.
Invalid address for the return parameter was specified.

Parent

See also

Property: `Parent` as `Dialogs` (read-only)

Description
Access the parent of the object.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowAnnotations

See also

Property: `ShowAnnotations` as `Boolean`

Description
Set this property to `true`, to show the annotations to a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowAttributes

See also

Property: `ShowAttributes` as `Boolean`

Description
Set this property to `true`, to show the type definitions attributes in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

© 2005 Altova GmbH

ShowChildren

See also

Property: `ShowChildren` as `Boolean`

Description
Set this property to `true`, to show the children of a type definition as links in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowDiagram

See also

Property: `ShowDiagram` as `Boolean`

Description
Set this property to `true`, to show type definitions as diagrams in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowEnumerations

See also

Property: `ShowEnumerations` as `Boolean`

Description
Set this property to `true`, to show the enumerations contained in a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowIdentityConstraints

See also

Property: `ShowIdentityConstraints` as `Boolean`

Description
Set this property to `true`, to show a type definitions identity constraints in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowNamespace

See also

Property: `ShowNamespace` as `Boolean`

Description
Set this property to `true`, to show the namespace of type definitions in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowPatterns

See also

Property: `ShowPatterns` as `Boolean`

Description
Set this property to `true`, to show the patterns of a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors
2900 The object is no longer valid.
2901 Invalid address for the return parameter was specified.

ShowProgressBar

See also

Property: `ShowProgressBar` as `Boolean`

Description
Set this property to `true`, to make the window showing the document generation progress

© 2005 Altova GmbH

visible. Use `false`, to hide it. Default is `false`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowProperties

See also

Property: `ShowProperties` as `Boolean`

Description

Set this property to `true`, to show the type definition properties in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowResult

See also

Property: `ShowResult` as `Boolean`

Description

Set this property to `true`, to automatically open the resulting document when generation was successful. HTML documentation will be opened in XMLSpy 2005. To show Word documentation, MS-Word will be started. The property gets initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowSingleFacets

See also

Property: `ShowSingleFacets` as `Boolean`

Description

Set this property to `true`, to show the facets of a type definition in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowSourceCode

See also

Property: `ShowSourceCode` as `Boolean`

Description
Set this property to `true`, to show the XML source code for type definitions in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowType

See also

Property: `ShowType` as `Boolean`

Description
Set this property to `true`, to show the type of type definitions in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

ShowUsedBy

See also

Property: `ShowUsedBy` as `Boolean`

Description
Set this property to `true`, to show the used-by relation for type definitions in the schema documentation. The property is initialized with the value used during the last call to `Document.GenerateSchemaDocumentation`.

Errors

2900 The object is no longer valid.

2901 Invalid address for the return parameter was specified.

© 2005 Altova GmbH

4.3.16 SpyProject

See also

Methods
`CloseProject`
`SaveProject`
`SaveProjectAs`

Properties
`RootItems`
`ProjectFile`

Description
`SpyProject` Class

CloseProject

See also

Declaration: `CloseProject(bDiscardChanges` as Boolean, `bCloseFiles` as Boolean, `bDialog` as Boolean)

Parameters
`bDiscardChanges`
Set `bDiscardChanges` to FALSE if you want to save the changes of the open project files and the project.

`bCloseFiles`
Set `bCloseFiles` to TRUE to close all open project files.

`bDialog`
Show dialogs for user input.

Description
`CloseProject` closes the current project.

ProjectFile

See also

Declaration: `ProjectFile` as String

Description
Path and filename of the project.

RootItems

See also

Declaration: `RootItems` as `SpyProjectItems`

Description
Root level of collection of project items.

SaveProject

See also

Declaration: `SaveProject`

Description
`SaveProject` saves the current project.

SaveProjectAs

See also

Declaration: `SaveProjectAs` (`strPath` as String, `bDialog` as Boolean)

Parameters
`strPath`
Full path with file name of new project file.

`bDialog`
If `bDialog` is TRUE, a file-dialog will be displayed.

Description
`SaveProjectAs` stores the project data into a new location.

4.3.17 SpyProjectItem

See also

Methods
`Open`

Properties
`ChildItems`
`ParentItem`
`FileExtensions`
`ItemType`
`Name`
`Path`
`ValidateWith`
`XMLForXSLTransformation`
`XSLForXMLTransformation`
`XSLTransformationFileExtension`
`XSLTransformationFolder`

Description
`SpyProjectItem` Class

ChildItems

See also

Declaration: `ChildItems` as `SpyProjectItems`

Description

© 2005 Altova GmbH

If the item is a folder, `ChildItems` is the collection of the folder content.

FileExtensions

See also

Declaration: `FileExtensions` as String

Description
Used to set the file extensions if the project item is a folder.

ItemType

See also

Declaration: `ItemType` as `SPYProjectItemTypes`

Description
This property is read-only.

Name

See also

Declaration: `Name` as String

Description
Name of the project item. This property is read-only.

Open

See also

Declaration: `Open` as `Document`

Return Value
The project item opened as document.

Description
Opens the project item.

ParentItem

See also

Declaration: `ParentItem` as `SpyProjectItem`

Description
Parent item of the current project item. Can be NULL (Nothing) if the project item is a top-level item.

Path

See also

Declaration: `Path` as String

Description
Path of project item. This property is read-only.

ValidateWith

See also

Declaration: `ValidateWith` as String

Description
Used to set the schema/DTD for validation.

XMLForXSLTransformation

See also

Declaration: `XMLForXSLTransformation` as String

Description
Used to set the XML for XSL transformation.

XSLForXMLTransformation

See also

Declaration: `XSLForXMLTransformation` as String

Description
Used to set the XSL for XML transformation.

XSLTransformationFileExtension

See also

Declaration: `XSLTransformationFileExtension` as String

Description
Used to set the file extension for XSL transformation output files.

XSLTransformationFolder

See also

Declaration: `XSLTransformationFolder` as String

Description
Used to set the destination folder for XSL transformation output files.

© 2005 Altova GmbH

4.3.18 SpyProjectItems

See also

Methods
AddFile
AddFolder
AddURL
RemoveItem

Properties
Count
Item

Description
SpyProjectItems Class

AddFile

See also

Declaration: AddFile (*strPath* as String)

Parameters

strPath
Full path with file name of new project item

Description
The method adds a new file to the collection of project items.

AddFolder

See also

Declaration: AddFolder (*strName* as String)

Parameters

strName
Name of the new folder.

Description
The method AddFolder adds a folder with the name strName to the collection of project items.

AddURL

See also

Declaration: AddURL (*strURL* as String, *nURLType* as SPYURLTypes, *strUser* as String, *strPassword* as String, *bSave* as Boolean)

Description

`strURL`
URL to open as document.

`nURLType`
Type of document to open. Set to -1 for auto detection.

`nLoading`
Set `nLoading` to 0 (zero) if you want to load it from cache or proxy. Otherwise set `nLoading` to 1.

`strUser`
Name of the user if required. Can be empty.

`strPassword`
Password for authentification. Can be empty.

`bSave`
Save user and password information.

Description
The method adds an URL item to the project collection.

Count

See also

Declaration: `Count` as long

Description
This property gets the count of project items in the collection. The property is read-only.

Item

See also

Declaration: `Item` (*n* as long) as `SpyProjectItem`

Description
Retrieves the n-th element of the collection of project items. The first item has index 1.

RemoveItem

See also

Declaration: `RemoveItem` (*`pItem`* as `SpyProjectItem`)

Description
RemoveItem deletes the item pItem from the collection of project items.

4.3.19 TextImportExportSettings

See also

Properties for import only
`ImportFile`

 © 2005 Altova GmbH

Properties for export only
`DestinationFolder`
`FileExtension`

Properties for import and export
`HeaderRow`
`FieldDelimiter`
`EnclosingCharacter`
`Encoding`
`EncodingByteOrder`

Description
`TextImportExportSettings` contains options common to text import and export functions.

DestinationFolder

See also

Property: `DestinationFolder` as String

Description
The property `DestinationFolder` sets the folder where the created files are saved during text export.

EnclosingCharacter

See also

Property: `EnclosingCharacter` as `SPYTextEnclosing`

Description
This property defines the character that encloses all field values for import and export. Default is `spyNoEnclosing`.

Encoding

See also

Property: `Encoding` as String

Description
The property `Encoding` sets the character encoding for the text files for importing and exporting.

EncodingByteOrder

See also

Property: `EncodingByteOrder` as `SPYEncodingByteOrder`

Description
The property `EncodingByteOrder` sets the byte order for Unicode characters. Default is `spyNONE`.

FieldDelimiter

See also

Property: `FieldDelimiter` as `SPYTextDelimiters`

Description
The property `FieldDelimiter` defines the delimiter between the fields during import and export. Default is `spyTabulator`.

FileExtension

See also

Property: `FileExtension` as String

Description
This property sets the file extension for files created on text export.

HeaderRow

See also

Property: `HeaderRow` as Boolean

Description
The property `HeaderRow` is used during import and export. Set `HeaderRow` true on import, if the first line of the text file contains the names of the columns. Set `HeaderRow` true on export, if the first line in the created text files should contain the name of the columns. Default value is true.

ImportFile

See also

Property: `ImportFile` as String

Description
This property is used to set the text file for import. The string has to be a full qualified path. See also Import and Export.

4.3.20 XMLData

See also

Properties
`Kind`
`Name`
`TextValue`

`HasChildren`
`MayHaveChildren`
`Parent`

© 2005 Altova GmbH

Methods
GetFirstChild
GetNextChild
GetCurrentChild

InsertChild
AppendChild

EraseAllChildren
EraseCurrentChild

IsSameNode

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind spyXMLDataElement with the Name property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Objects of this class represent the different atomic parts of an XML document. See the enumeration type SPYXMLDataKind for the available part types. Each part knows its children, thus forming a XMLData tree with Document.RootElement at its top. To get the top element of the document content - ignoring the XML header - use Document.DataRoot. For examples on how to traverse the XMLData tree see Using XMLData to modify document structure and GetNextChild.

AppendChild

See also

Declaration: AppendChild (*pNewData* as XMLData)

Description
AppendChild appends pNewData as last child to the XMLData object. See also Using XMLData.

Errors

1500	The XMLData object is no longer valid.
1505	Invalid XMLData kind was specified.
1506	Invalid address for the return parameter was specified.
1507	Element cannot have Children
1512	Cyclic insertion - new data element is already part of document
1900	Document must not be modified

Example

```
Dim objCurrentParent As XMLData
```

```
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpy.ActiveDocument.RootElement

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

EraseAllChildren

See also

Declaration: `EraseAllChildren`

Description
`EraseAllChildren` deletes all associated children of the `XMLData` object.

Errors

1500 The XMLData object is no longer valid.

1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

EraseCurrentChild

See also

Declaration: `EraseCurrentChild`

Description
`EraseCurrentChild` deletes the current `XMLData` child object. Before you call `EraseCurrentChild` you must initialize an internal iterator with `XMLData.GetFirstChild`. After deleting the current child, `EraseCurrentChild` increments the internal iterator of the XMLData element. No error is returned when the last child gets erased and the iterator is moved past the end of the child list. The next call to `EraseCurrentChild` however, will return error 1503.

Errors

1500 The XMLData object is no longer valid.

1503 No iterator is initialized for this XMLData object, or the iterator points past the last child.

1900 Document must not be modified

Examples

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ---------------------------------------
// let's get an XMLData element, we assume that the
```

 © 2005 Altova GmbH

```
// cursor selects the parent of a list in grid view
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// the following line would be shorter, of course
//objList.EraseAllChildren ();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
 try
 {
  objEle = objList.GetFirstChild(-1);
  while (objEle != null)
   objList.EraseCurrentChild();
   // no need to call GetNextChild
 }
 catch (err)
  // 1503 - we reached end of child list
  { if ((err.number & 0xffff) != 1503) throw (err); }
}
```

GetCurrentChild

See also

Declaration: `GetCurrentChild` as `XMLData`

Return Value
Returns an XML element as `XMLData` object.

Description
`GetCurrentChild` gets the current child. Before you call `GetCurrentChild` you must initialize an internal iterator with `XMLData.GetFirstChild`.

Errors

1500	The XMLData object is no longer valid.
1503	No iterator is initialized for this XMLData object.
1510	Invalid address for the return parameter was specified.

GetFirstChild

See also

Declaration: `GetFirstChild` (*`nKind`* as `SPYXMLDataKind`) as `XMLData`

Return Value
Returns an XML element as `XMLData` object.

Description
`GetFirstChild` initializes a new iterator and returns the first child. Set `nKind` = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use `GetCurrentChild`, `GetNextChild` or `EraseCurrentChild`.

Errors

1500 The XMLData object is no longer valid.

1501 Invalid XMLData kind was specified.

1504 Element has no children of specified kind.

1510 Invalid address for the return parameter was specified.

Example

See the example at `XMLData.GetNextChild`.

GetNextChild

See also

Declaration: `GetNextChild` as `XMLData`

Return Value

Returns an XML element as `XMLData` object.

Description

`GetNextChild` steps to the next child of this element. Before you call `GetNextChild` you must initialize an internal iterator with `XMLData.GetFirstChild`.

Check for the last child of the element as shown in the sample below.

Errors

1500 The XMLData object is no longer valid.

1503 No iterator is initialized for this XMLData object.

1510 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' VBA code snippet - iterate XMLData children
' ---------------------------------------------
On Error Resume Next
Set objParent = objSpy.ActiveDocument.RootElement

'get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)

Do
  'do something useful with the child

  'step to next child
  Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ---------------------------------------
try
{
   var objXMLData = ... // initialize somehow
   var objChild = objXMLData.GetFirstChild(-1);
```


© 2005 Altova GmbH

```
   while (true)
   {
      // do something usefull with objChild

      objChild = objXMLData.GetNextChild();
   }
}
catch (err)
{
   if ((err.number & 0xffff) == 1504)
    ; // element has no children
   else if ((err.number & 0xffff) == 1503)
    ;// last child reached
   else
    throw (err);
}
```

HasChildren

See also

Declaration: `HasChildren` as Boolean

Description
The property is true if the object is the parent of other `XMLData` objects. This property is read-only.

Errors

1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

InsertChild

See also

Declaration: `InsertChild` (*`pNewData`* as `XMLData`)

Description
`InsertChild` inserts the new child before the current child (see also `XMLData.GetFirstChild`, `XMLData.GetNextChild` to set the current child).

Errors

1500 The XMLData object is no longer valid.

1503 No iterator is initialized for this XMLData object.

1505 Invalid XMLData kind was specified.

1506 Invalid address for the return parameter was specified.

1507 Element cannot have Children

1512 Cyclic insertion - new data element is already part of document

1900 Document must not be modified

Examples

See Using XMLData to modify document structure.

IsSameNode

See also

Declaration: IsSameNode (*pNodeToCompare* as XMLData) as Boolean

Description

Returns true if pNodeToCompare references the same node as the object itself.

Errors

1500 The XMLData object is no longer valid.

1506 Invalid address for the return parameter was specified.

Kind

See also

Declaration: Kind as SPYXMLDataKind

Description
Kind of this XMLData object. This property is read-only.

Errors

1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

MayHaveChildren

See also

Declaration: MayHaveChildren as Boolean

Description
Indicates whether it is allowed to add children to this XMLData object.
This property is read-only.

Errors

1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

Name

See also

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

 © 2005 Altova GmbH

Errors

1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

Parent

See also

Declaration: `Parent` as `XMLData`

Return value
Parent as `XMLData` object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors

1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

TextValue

See also

Declaration: `TextValue` as String

Description
Used to modify and to get the text value of this `XMLData` object.

Errors

1500 The XMLData object is no longer valid.

1510 Invalid address for the return parameter was specified.

Examples
See `Using XMLData to modify document structure`.

4.4 Interfaces (obsolete)

Interfaces contained in this book are obsolete. It is recommended to migrate your applications to the new interfaces. See the different properties and methods in this book for migration hints.

4.4.1 AuthenticEvent (obsolete)

Superseded by AuthenticView and **AuthenticRange**

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties

```
altKey
altLeft
ctrlKey
ctrlLeft
shiftKey
shiftLeft

keyCode
repeat

button

clientX
clientY

dataTransfer

srcElement
fromElement

propertyName

cancelBubble
returnValue

type
```

Description

© 2005 Altova GmbH

`DocEditEvent` interface.

altKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//        MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

See also

Declaration: `altKey` as Boolean

Description
True if the right ALT key is pressed.

altLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.altKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//        MsgBox ("alt key is down");
// }
// use now:
function On_AuthenticView_KeyDown (SPYKeyEvent i_eKeyEvent, long i_nKeyCode,
SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nVirtualKeyStatus & spyAltKeyMask)
        MsgBox ("alt key is down");
}
```

See also

Declaration: `altLeft` as Boolean

Description
True if the left ALT key is pressed.

© 2005 Altova GmbH

button (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditButtonDown ()
// {
//     if (Application.ActiveDocument.DocEditView.event.button == 1)
//         MsgBox ("left mouse button down detected");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyLeftButtonDownMask)
        MsgBox ("left mouse button down detected");
}
```

See also

Declaration: `button` as long

Description
Specifies which mouse button is pressed:

0	No button is pressed.
1	Left button is pressed.
2	Right button is pressed.
3	Left and right buttons are both pressed.
4	Middle button is pressed.
5	Left and middle buttons both are pressed.
6	Right and middle buttons are both pressed.
7	All three buttons are pressed.

cancelBubble (obsolete)

Superseded by the boolean return value of following event handler functions
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

Returning *true* from an event handler function signals that the event has beend handled and normal event handling should be aborted.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//        // cancel key processing, swallow spaces :-)
//        Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

See also

Declaration: `cancelBubble` as Boolean

Description
Set `cancelBubble` to TRUE if the default event handler should not be called.

© 2005 Altova GmbH

clientX (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " +
Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

See also

Declaration: `clientX` as long

Description
X value of the current mouse position in client coordinates.

clientY (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     MsgBox ("moving over " +
Application.ActiveDocument.DocEditView.event.clientX +
//             "/" + Application.ActiveDocument.DocEditView.event.clientY);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyMouseMoveMask)
        MsgBox ("moving over " + i_nXPos + "/" + i_nYPos);
}
```

See also

Declaration: `clientY` as long

Description
Y value of the current mouse position in client coordinates.

© 2005 Altova GmbH

ctrlKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//        MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

See also

Declaration: `ctrlKey` as Boolean

Description
True if the right CTRL key is pressed.

ctrlLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     if (Application.ActiveDocument.DocEditView.event.ctrlKey ||
//         Application.ActiveDocument.DocEditView.event.altLeft)
//        MsgBox ("control key is down");
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if (i_eMouseEvent & spyCtrlKeyMask)
        MsgBox ("control key is down");
}
```

See also

Declaration: `ctrlLeft` as Boolean

Description
True if the left CTRL key is pressed.

© 2005 Altova GmbH

dataTransfer (obsolete)

Superseded by parameters to
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDrop ()
// {
//     if (Application.ActiveDocument.DocEditView.event.dataTransfer !=
null)
//         if (!
Application.ActiveDocument.DocEditView.event.dataTransfer.ownDrag)
//         {
//            // cancel key processing, don't drop foreign objects :-)
//            Application.ActiveDocument.DocEditView.event.cancelBubble =
true;
//         }
// }
// use now:
function On_AuthenticView_BeforeDrop (long i_nXPos, long i_nYPos,
                                      IAuthenticRange *i_ipRange,
                                      IAuthenticDataTransfer *i_ipData)
{
    if (i_ipRange != null)
        if (! i_ipRange.ownDrag)
            return true;  // cancel key processing, don't drop foreign
objects :-)

    return false;
}
```

See also

Declaration: `dataTransfer` as Variant

Description
Property `dataTransfer`.

fromElement (obsolete)

Not supported

See also

Declaration: `fromElement` as Variant (not supported)

Description
Currently no event sets this property.

keyCode (obsolete)

Superseded by a parameter to AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditKeyPressed ()
// {
//     if (Application.ActiveDocument.DocEditView.event.keyCode == 0x20)
//     {
//        // cancel key processing, swallow spaces :-)
//        Application.ActiveDocument.DocEditView.event.cancelBubble = true;
//     }
// }
// use now:
function On_AuthenticView_KeyPressed (SPYKeyEvent i_eKeyEvent, long
i_nKeyCode, SPYVirtualKeyMask i_nVirtualKeyStatus)
{
    if (i_nKeyCode == 0x20)
        return true;  // cancel key processing, swallow spaces :-)
}
```

See also

Declaration: `keyCode` as long

Description
Keycode of the currently pressed key. This property is read-write.

© 2005 Altova GmbH

propertyName (obsolete)

Not supported

See also

Declaration: `propertyName` as String (not supported)

Description
Currently no event sets this property.

repeat (obsolete)

Not supported

See also

Declaration: `repeat` as Boolean (not supported)

Description
True if the `onkeydown` event is repeated.

returnValue (obsolete)

No longer supported

See also

Declaration: `returnValue` as Variant

Description
Use `returnValue` to set a return value for your event handler.

shiftKey (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//        MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
    if (i_eMouseEvent & spyShiftKeyMask)
        MsgBox ("shift key is down");
}
```

See also

Declaration: `shiftKey` as Boolean

Description
True if the right SHIFT key is pressed.

© 2005 Altova GmbH

shiftLeft (obsolete)

Superseded by parameters to
AuthenticView.OnKeyboardEvent (On_AuthenticView_KeyPressed)
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditDragOver ()
// {
//     if (Application.ActiveDocument.DocEditView.event.shiftKey ||
//         Application.ActiveDocument.DocEditView.event.shiftLeft)
//        MsgBox ("shift key is down");
// }
// use now:
function On_AuthenticView_DragOver (long i_nXPos, long i_nYPos,
                                    SPYMouseEvent i_eMouseEvent,
                                    IAuthenticRange *i_ipRange,
                                    IAuthenticDataTransfer *i_ipData)
{
    if (i_eMouseEvent & spyShiftKeyMask)
        MsgBox ("shift key is down");
}
```

See also

Declaration: `shiftLeft` as Boolean

Description
True if the left SHIFT key is pressed.

srcElement (obsolete)

Superseded by parameters to
AuthenticView.OnMouseEvent (On_AuthenticView_MouseEvent)
AuthenticView.OnBeforeDrop (On_AuthenticView_BeforeDrop)
AuthenticView.OnDragOver (On_AuthenticView_DragOver)

The event object that holds the information of the last event is now replaced by parameters to the different event handler functions to simplify data access. The event object will be supported for a not yet defined period of time for compatibility reasons. No improvements are planned. It is highly recommended to migrate to the new event handler functions.

With the new event handler function, a range object selecting this element is provided instead of the XMLData element currently below the mouse cursor.

```
// ----- XMLSpy scripting environment - javascript sample -----
// instead of:
// function On_DocEditMouseMove ()
// {
//     var objEvent = Application.ActiveDocument.DocEditView.event;
//     if (objEvent.srcElement != null)
//         MsgBox ("moving over " + objEvent.srcElement.Parent.Name);
// }
// use now:
function On_AuthenticView_MouseEvent (long i_nXPos, long i_nYPos,
SPYMouseEvent i_eMouseEvent, IAuthenticRange *i_ipRange)
{
    if ((i_eMouseEvent & spyMouseMoveMask) &&
        (i_ipRange != null))
        MsgBox ("moving over " + i_ipRange.FirstXMLData.Parent.Name);
}
```

See also

Declaration: `srcElement` as Variant

Description
Element which fires the current event. This is usually an `XMLData` object.

© 2005 Altova GmbH

type (obsolete)

Not supported

See also

Declaration: `type` as String (not supported)

Description
Currently no event sets this property.

4.4.2 AuthenticSelection (obsolete)

Superseded by AuthenticRange

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interface. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Properties
`Start`
`StartTextPosition`
`End`
`EndTextPosition`

© 2005 Altova GmbH

End (obsolete)

Superseded by AuthenticRange.LastXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData =
Application.ActiveDocument.DocEditView.CurrentSelection.End;
// use now:
var objXMLData =
Application.ActiveDocument.AuthenticView.Selection.LastXMLData;
```

See also

Declaration: `End` as `XMLData`

Description
XML element where the current selection ends.

EndTextPosition (obsolete)

Superseded by AuthenticRange.LastXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset =
Application.ActiveDocument.DocEditView.CurrentSelection.EndTextPosition;
// use now:
var nOffset =
Application.ActiveDocument.AuthenticView.Selection.LastXMLDataOffset;
```

See also

Declaration: `EndTextPosition` as long

Description
Position in `DocEditSelection.End.TextValue` where the selection ends.

Start (obsolete)

Superseded by AuthenticRange.FirstXMLData

```
// ----- javascript sample -----
// instead of:
// var objXMLData =
Application.ActiveDocument.DocEditView.CurrentSelection.Start;
// use now:
var objXMLData =
Application.ActiveDocument.AuthenticView.Selection.FirstXMLData;
```

See also

Declaration: `Start` as `XMLData`

Description
XML element where the current selection starts.

StartTextPosition (obsolete)

Superseded by AuthenticRange.FirstXMLDataOffset

```
// ----- javascript sample -----
// instead of:
// var nOffset =
Application.ActiveDocument.DocEditView.CurrentSelection.StartTextPosition;
// use now:
var nOffset =
Application.ActiveDocument.AuthenticView.Selection.FirstXMLDataOffset;
```

See also

Declaration: `StartTextPosition` as long

Description
Position in `DocEditSelection.Start.TextValue` where the selection starts.

© 2005 Altova GmbH

4.4.3 OldAuthentictView (obsolete)

Superseded by AuthenticView and **AuthenticRange**

The DocEditView object is renamed to OldAuthenticView.
DocEditSelection is renamed to AuthenticSelection.
DocEditEvent is renamed to AuthenticEvent.
DocEditDataTransfer is renamed to AuthenticDataTransfer.

Their usage - except for AuthenticDataTransfer - is no longer recommended. We will continue to support existing functionality for a yet undefined period of time but no new features will be added to these interfaces. All functionality available up to now in DocEditView, DocEditSelection, DocEditEvent and DocEditDataTransfer is now available via AuthenticView, AuthenticRange and AuthenticDataTransfer. Many new features have been added.

For examples on migrating from DocEdit to Authentic see the description of the different methods and properties of the different DocEdit objects.

See also

Methods

```
LoadXML
SaveXML

EditClear
EditCopy
EditCut
EditPaste
EditRedo
EditSelectAll
EditUndo

RowAppend
RowDelete
RowDuplicate
RowInsert
RowMoveDown
RowMoveUp

ApplyTextState
IsTextStateApplied
IsTextStateEnabled

MarkUpView

SelectionSet
SelectionMoveTabOrder

GetNextVisible
GetPreviousVisible

GetAllowedElements
```

Properties

```
CurrentSelection

event

XMLRoot

IsEditClearEnabled
IsEditCopyEnabled
IsEditCutEnabled
IsEditPasteEnabled
IsEditRedoEnabled
IsEditUndoEnabled

IsRowAppendEnabled
IsRowDeleteEnabled
IsRowDuplicateEnabled
IsRowInsertEnabled
IsRowMoveDownEnabled
IsRowMoveUpEnabled
```

Description

Interface for Authentic.

© 2005 Altova GmbH

ApplyTextState (obsolete)

Superseded by AuthenticRange.PerformAction

Use spyAuthenticApply for the eAction parameter. The PerformAction method allows to apply text state attributes to any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.ApplyTextState ("bold");
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.PerformAction
(spyAuthenticApply, "bold"))
    MsgBox ("Error: can't set current selection to bold");
```

See also

Declaration: `ApplyTextState` (*`elementName`* as String)

Description
Applies or removes the text state defined by the parameter `elementName`. Common examples for the parameter elementName would be strong and italic.

In an XML document there are segments of data, which may contain sub-elements. For example consider the following HTML:

```
<b>fragment</b>
```

The HTML tag <b> will cause the word fragment to be bold. However, this only happens because the HTML parser knows that the tag <b> is bold. With XML there is much more flexibility. It is possible to define any XML tag to do anything you desire. The point is that it is possible to apply a Text state using XML. But the Text state that is applied must be part of the schema. For example in the `OrgChart.xml, OrgChart.sps, OrgChart.xsd` example the tag `<strong>` is the same as bold. And to apply bold the method `ApplyTextState()` is called. But like the row and edit operations it is necessary to test if it is possible to apply the text state.

See also `IsTextStateEnabled` and `IsTextStateApplied`.

CurrentSelection (obsolete)

Superseded by AuthenticView.Selection

The returned AuthenticRange object supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// var objDocEditSel =
Application.ActiveDocument.DocEditView.CurrentSelection;
// use now:
    var objRange = Application.ActiveDocument.AuthenticView.Selection;
```

See also

Declaration: `CurrentSelection` as `DocEditSelection`

Description
The property provides access to the current selection in the Authentic.

EditClear (obsolete)

Superseded by AuthenticRange.Delete

The Delete method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditClear();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Delete())
    MsgBox ("Error: can't delete current selection");
```

See also

Declaration: `EditClear`

Description
Deletes the current selection.

© 2005 Altova GmbH

EditCopy (obsolete)

Superseded by AuthenticRange.Copy

The Copy method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCopy();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Copy())
    MsgBox ("Error: can't copy current selection");
```

See also

Declaration: `EditCopy`

Description
Copies the current selection to the clipboard.

EditCut (obsolete)

Superseded by AuthenticRange.Cut

The Cut method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditCut();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Cut())
    MsgBox ("Error: can't cut out current selection");
```

See also

Declaration: `EditCut`

Description
Cuts the current selection from the document and copies it to the clipboard.

EditPaste (obsolete)

Superseded by AuthenticRange.Paste

The Paste method of AuthenticRange allows to delete any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditPaste();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.Paste())
    MsgBox ("Error: can't paste to current selection");
```

See also

Declaration: `EditPaste`

Description
Pastes the content from the clipboard into the document.

EditRedo (obsolete)

Superseded by AuthenticView.Redo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Redo())
    MsgBox ("Error: no redo step available");
```

See also

Declaration: `EditRedo`

Description
Redo the last undo step.

© 2005 Altova GmbH

EditSelectAll (obsolete)

Superseded by AuthenticView.WholeDocument and **AuthenticRange.Select**

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditSelectAll();
// use now:
Application.ActiveDocument.AuthenticView.WholeDocument.Select();
```

See also

Declaration: `EditSelectAll`

Description
The method selects the complete document.

EditUndo (obsolete)

Superseded by AuthenticView.Undo

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (! Application.ActiveDocument.AuthenticView.Undo())
    MsgBox ("Error: no undo step available");
```

See also

Declaration: `EditUndo`

Description
Undo the last action.

event (obsolete)

Superseded by parameters to AuthenticView events.

See also

Declaration: `event` as `DocEditEvent`

Description
The event property holds a `DocEditEvent` object which contains information about the current event.

GetAllowedElements (obsolete)

Superseded by AuthenticRange.CanPerformActionWith

AuthenticRange now supports all functionality of the 'elements' entry helper. Besides querying the elements that can be inserted, appended, etc., you can invoke the action as well. See AuthenticRange.PerformAction for more information.

```
// ----- javascript sample -----
// instead of:
// var arrElements = New Array();
// var objDocEditView = Application.ActiveDocument.DocEditView;
// var objStartElement = objDocEditView.CurrentSelection.Start;
// var objEndElement = objDocEditView.CurrentSelection.End;
// objDocEditView.GetAllowedElements(k_ActionInsertBefore, objStartElement,
objEndElement, arrElements);
// use now:
var arrElements = New Array();
Application.ActiveDocument.AuthenticView.Selection.CanPerformActionWith
(spyAuthenticInsertBefore, arrElements);
```

See also

Declaration: `GetAllowedElements` (*`nAction`* as `SpyAuthenticElementActions`, *`pStartElement`* as `XMLData`, *`pEndElement`* as `XMLData`, *`pElements`* as Variant)

Description
`GetAllowedElements()` returns the allowed elements for the various actions specified by `nAction`.

JavaScript example:

```
Function GetAllowed()
{
 var objView = Application.ActiveDocument.DocEditView;

 var arrElements = New Array(1);
```


© 2005 Altova GmbH

```
 var objStart = objView.CurrentSelection.Start;
 var objEnd = objView.CurrentSelection.End;

 var strText;
 strText = "valid elements at current selection:\n\n";

 For(var i = 1;i <= 4;i++){
  objPlugIn.GetAllowedElements(i,objStart,objEnd,arrElements);
  strText = strText + ListArray(arrElements) + "------------------\n";
 }

 Return strText;
}

Function ListArray(arrIn)
{
 var strText = "";

 If(TypeOf(arrIn) == "object"){
  For(var i = 0;i <= (arrIn.length - 1);i++)
   strText = strText + arrIn[i] + "\n";
 }

 Return strText;
}
```

VBScript example:

```
Sub DisplayAllowed
 Dim objView
 Set objView = Application.ActiveDocument.DocEditView

 Dim arrElements()

 Dim objStart
 Dim objEnd
 Set objStart = objView.CurrentSelection.Start
 Set objEnd = objView.CurrentSelection.End

 Dim strText
 strText = "valid elements at current selection:" & chr(13) & chr(13)

 Dim i

 For i = 1 To 4
  objView.GetAllowedElements i,objStart,objEnd,arrElements
  strText = strText & ListArray(arrElements) & "---------------" & chr(13)
 Next

 msgbox strText
End Sub

Function ListArray(arrIn)
 Dim strText

 If IsArray(arrIn) Then
  Dim i

  For i = 0 To UBound(arrIn)
   strText = strText & arrIn(i) & chr(13)
  Next
 End If

 ListArray = strText
End Function
```

© 2005 Altova GmbH

GetNextVisible (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the next XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData =
Application.ActiveDocument.DocEditView.GetNextVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0,
objXMLData, -1);
// use now:
var objRange = ...
try
    { objRange.SelectNext (spyAuthenticTag).Select(); }
catch (err)
{
    if ((err.number & 0xffff) == 2003)
        MsgBox ("end of document reached");
    else
        throw (err);
}
```

See also

Declaration: `GetNextVisible` (*`pElement`* as `XMLData`) as `XMLData`

Description
The method gets the next visible XML element in the document.

GetPreviousVisible (obsolete)

Superseded by AuthenticRange.SelectPrevious

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the text passage that represents the content of the previous XML element is just one of them.

```
// ----- javascript sample -----
// instead of:
// var objCurrXMLData = ...
// var objXMLData =
Application.ActiveDocument.DocEditView.GetPreviousVisible(objCurrXMLData);
// Application.ActiveDocument.DocEditView.SelectionSet (objXMLData, 0,
objXMLData, -1);
// use now:
var objRange = ...
try
    { objRange.SelectPrevious (spyAuthenticTag).Select(); }
catch (err)
{
    if ((err.number & 0xffff) == 2004)
        MsgBox ("begin of document reached");
    else
        throw (err);
}
```

See also

Declaration: `GetPreviousVisible` (*`pElement`* as `XMLData`) as `XMLData`

Description
The method gets the previous visible XML element in the document.

© 2005 Altova GmbH

IsEditClearEnabled (obsolete)

Superseded by AuthenticRange.IsDeleteEnabled

The IsDeleteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditClearEnabled)
//    Application.ActiveDocument.DocEditView.EditClear();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsDeleteEnabled)
    objCurrSelection.Delete();
```

See also

Declaration: `IsEditClearEnabled` as Boolean

Description
True if `EditClear` is possible. See also `Editing operations`.

IsEditCopyEnabled (obsolete)

Superseded by AuthenticRange.IsCopyEnabled

The IsCopyEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCopyEnabled)
//    Application.ActiveDocument.DocEditView.EditCopy();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCopyEnabled)
    objCurrSelection.Copy();
```

See also

Declaration: `IsEditCopyEnabled` as Boolean

Description
True if copy to clipboard is possible. See also `EditCopy` and `Editing operations`.

IsEditCutEnabled (obsolete)

Superseded by AuthenticRange.IsCutEnabled

The IsCutEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditCutEnabled)
//    Application.ActiveDocument.DocEditView.EditCut();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsCutEnabled)
    objCurrSelection.Cut();
```

See also

Declaration: `IsEditCutEnabled` as Boolean

Description
True if `EditCut` is currently possible. See also `Editing operations`.

IsEditPasteEnabled (obsolete)

Superseded by AuthenticRange.IsPasteEnabled

The IsPasteEnabled property is now supported for any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditPasteEnabled)
//    Application.ActiveDocument.DocEditView.EditPaste();
// use now:
var objCurrSelection = Application.ActiveDocument.AuthenticView.Selection;
if (objCurrSelection.IsPasteEnabled)
    objCurrSelection.Paste();
```

See also

Declaration: `IsEditPasteEnabled` as Boolean

Description
True if `EditPaste` is possible. See also `Editing operations`.

© 2005 Altova GmbH

IsEditRedoEnabled (obsolete)

Superseded by AuthenticView.IsRedoEnabled

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditRedoEnabled)
//    Application.ActiveDocument.DocEditView.EditRedo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsRedoEnabled)
    Application.ActiveDocument.AuthenticView.Redo();
```

See also

Declaration: `IsEditRedoEnabled` as Boolean

Description
True if `EditRedo` is currently possible. See also `Editing operations`.

IsEditUndoEnabled (obsolete)

Superseded by AuthenticView.IsUndoEnabled

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsEditUndoEnabled)
//    Application.ActiveDocument.DocEditView.EditUndo();
// use now:
if (Application.ActiveDocument.AuthenticView.IsUndoEnabled)
    Application.ActiveDocument.AuthenticView.Undo();
```

See also

Declaration: `IsEditUndoEnabled` as Boolean

Description
True if `EditUndo` is possible. See also `Editing operations`.

IsRowAppendEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowAppendEnabled)
//    Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.AppendRow();
```

See also

Declaration: `IsRowAppendEnabled` as Boolean

Description
True if RowAppend is possible. See also `Row operations`.

IsRowDeleteEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDeleteEnabled)
//    Application.ActiveDocument.DocEditView.Rowdelete();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DeleteRow();
```

See also

Declaration: `IsRowDeleteEnabled` as Boolean

Description
True if `RowDelete` is possible. See also `Row operations`.

© 2005 Altova GmbH

IsRowDuplicateEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowDuplicateEnabled)
//    Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.DuplicateRow();
```

See also

Declaration: `IsRowDuplicateEnabled` as Boolean

Description
True if `RowDuplicate` is currently possible. See also `Row operations`.

IsRowInsertEnabled (obsolete)

Superseded by AuthenticRange.IsInDynamicTable

The operations 'insert', 'append', 'delete' and 'duplicate' row are available whenever the selection is inside a dynamic table.

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowInsertEnabled)
//    Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsInDynamicTable())
    Application.ActiveDocument.AuthenticView.Selection.InsertRow();
```

See also

Declaration: `IsRowInsertEnabled` as Boolean

Description
True if `RowInsert` is possible. See also `Row operations`.

IsRowMoveDownEnabled (obsolete)

Superseded by AuthenticRange.IsLastRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.OldAuthenticView.IsRowMoveDownEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsLastRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowDown();
```

See also

Declaration: `IsRowMoveDownEnabled` as Boolean

Description
True if RowMoveDown is currently possible. See also `Row operations`.

IsRowMoveUpEnabled (obsolete)

Superseded by AuthenticRange.IsFirstRow

```
// ----- javascript sample -----
// instead of:
// if (Application.ActiveDocument.DocEditView.IsRowMoveUpEnabled)
//    Application.ActiveDocument.DocEditView.RowMoveUp();
// use now:
if (!Application.ActiveDocument.AuthenticView.Selection.IsFirstRow)
    Application.ActiveDocument.AuthenticView.Selection.MoveRowUp();
```

See also

Declaration: `IsRowMoveUpEnabled` as Boolean

Description
True if RowMoveUp is possible. See also `Row operations`.

© 2005 Altova GmbH

IsTextStateApplied (obsolete)

Superseded by AuthenticRange.IsTextStateApplied

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateApplied ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.IsTextStateApplied
("bold"))
    MsgBox ("bold on");
else
    MsgBox ("bold off");
```

See also

Declaration: `IsTextStateApplied` (*`elementName`* as String) as Boolean

Description
Checks to see if the it the text state has already been applied. Common examples for the parameter `elementName` would be strong and italic.

IsTextStateEnabled (obsolete)

Superseded by AuthenticRange.CanPerformAction

Use spyAuthenticApply for the eAction parameter. The CanPerformAction method allows to operate on any range of the document, not only the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.IsTextStateEnabled ("bold");
// use now:
if (Application.ActiveDocument.AuthenticView.Selection.CanPerformAction
(spyAuthenticApply, "bold"))
    ... // e.g. enable 'bold' button
```

See also

Declaration: `IsTextStateEnabled` (*`i_strElementName`* as String) as Boolean

Description
Checks to see if it is possible to apply a text state. Common examples for the parameter `elementName` would be strong and italic.

LoadXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property AsXMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.LoadXML (strDocAsXMLString);
// use now:
try
    { Application.ActiveDocument.AuthenticView.AsXMLString =
strDocAsXMLString; }
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

See also

Declaration: `LoadXML` (*`xmlString`* as String)

Description
Loads the current XML document with the XML string applied. The new content is displayed immediately.

MarkUpView (obsolete)

Superseded by AuthenticView.MarkupVisibility

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.MarkuUpView = 2;
// use now:
Application.ActiveDocument.AuthenticView.MarkupVisibility =
spyAuthenticMarkupLarge;
```

See also

Declaration: `MarkUpView` (*`kind`* as long)

Description
By default the document displayed is using HTML techniques. But sometimes it is desirable to show the editing tags. Using this method it is possible to display three different types of markup tags:

0	hide the markup tags
2	show the large markup tags
3	show the mixed markup tags.

© 2005 Altova GmbH

RowAppend (obsolete)

Superseded by AuthenticRange.AppendRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.AppendRow())
    MsgBox ("Error: can't append row");
```

See also

Declaration: `RowAppend`

Description
Appends a row at the current position.

See also Row operations.

RowDelete (obsolete)

Superseded by AuthenticRange.DeleteRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDelete();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DeleteRow())
    MsgBox ("Error: can't delete row");
```

See also

Declaration: `RowDelete`

Description
Deletes the currently selected row(s).

See also Row operations.

RowDuplicate (obsolete)

Superseded by AuthenticRange.DuplicateRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowDuplicate();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.DuplicateRow())
    MsgBox ("Error: can't duplicate row");
```

See also

Declaration: `RowDuplicate`

Description
The method duplicates the currently selected rows.

See also Row operations.

RowInsert (obsolete)

Superseded by AuthenticRange.InsertRow

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowInsert();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.InsertRow())
    MsgBox ("Error: can't insert row");
```

See also

Declaration: `RowInsert`

Description
Inserts a new row immediately above the current selection.

See also Row operations.

© 2005 Altova GmbH

RowMoveDown (obsolete)

Superseded by AuthenticRange.MoveRowDown

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowMoveDown();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowDown())
    MsgBox ("Error: can't move row down");
```

See also

Declaration: RowMoveDown

Description
Moves the current row one position down.

See also Row operations.

RowMoveUp (obsolete)

Superseded by AuthenticRange.MoveRowUp

The table operations of AuthenticRange now allow to manipulate any table in the current document independent of the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.RowAppend();
// use now:
if (! Application.ActiveDocument.AuthenticView.Selection.MoveRowUp())
    MsgBox ("Error: can't move row up");
```

See also

Declaration: RowMoveUp

Description
Moves the current row one position up.

See also Row operations.

SaveXML (obsolete)

Superseded by AuthenticView.AsXMLString

AuthenticView now supports the property XMLString that can be used to directly access and replace the document content as an XMLString.

```
// ----- javascript sample -----
// instead of:
// var strDocAsXMLString = Application.ActiveDocument.DocEditView.SaveXML();
// use now:
try
{
    var strDocAsXMLString =
Application.ActiveDocument.AuthenticView.AsXMLString;
    ... // do something here
}
catch (err)
    { MsgBox ("Error: invalid XML string"); }
```

See also

Declaration: SaveXML as String

Return Value
XML structure as string

Description
Saves the current XML data to a string that is returned to the caller.

© 2005 Altova GmbH

SelectionMoveTabOrder (obsolete)

Superseded by AuthenticRange.SelectNext

AuthenticRange now supports a wide range of element navigation methods based on document elements like characters, words, tags and many more. Selecting the next paragraph is just one of them, and navigation is not necessarily bound to the current UI selection.

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.DocEditView.SelectionMoveTabOrder(true, true);
// use now:
Application.ActiveDocument.AuthenticView.Selection.SelectNext
(spyAuthenticParagraph).Select();
// to append a row to a table use AuthenticRange.AppendRow
```

See also

Declaration: `SelectionMoveTabOrder` (*`bForward`* as Boolean, *`bTag`* as Boolean)

Description
`SelectionMoveTabOrder()` moves the current selection forwards or backwards.

If `bTag` is false and the current selection is at the last cell of a table a new line will be added.

SelectionSet (obsolete)

Superseded by AuthenticRange.FirstXMLData and related properties

AuthenticRange supports navigation via XMLData elements as well as navigation by document elements (e.g. characters, words, tags) or text cursor positions.

```
// ----- javascript sample -----
// instead of:
// if (! Application.ActiveDocument.DocEditView.SelectionSet(varXMLData1, 0,
varXMLData2, -1))
//    MsgBox ("Error: invalid data position");
// use now:
try
{
    var objSelection = Application.ActiveDocument.AuthenticView.Selection;
    objSelection.FirstXMLData = varXMLData1;
    objSelection.FirstXMLdataOffset = 0;
    objSelection.LastXMLData = varXMLData2;
    objSelection.LastXMLDataOffset = -1;
    objSelection.Select();
}
catch (err)
    { MsgBox ("Error: invalid data position"); }
// to select all text between varXMLData1 and varXMLdata2, inclusive
```

See also

Declaration: `SelectionSet` (*`pStartElement`* as `XMLData`, *`nStartPos`* as long, *`pEndElement`* as `XMLData`, *`nEndPos`* as long) as Boolean

Description
Use `SelectionSet()` to set a new selection in the Authentic. Its possible to set `pEndElement` to null (nothing) if the selection should be just over one (`pStartElement`) XML element.

© 2005 Altova GmbH

XMLRoot (obsolete)

Superseded by AuthenticView.XMLDataRoot

```
// ----- javascript sample -----
// instead of:
// var objXMLData = Application.ActiveDocument.DocEditView.XMLRoot;
// use now:
var objXMLData = Application.ActiveDocument.AuthenticView.XMLDataRoot;
```

See also

Declaration: `XMLRoot` as `XMLData`

Description
`XMLRoot` is the parent element of the currently displayed XML structure. Using the `XMLData` interface you have full access to the complete content of the file.

See also Using XMLData for more information.

© 2005 Altova GmbH

4.5 Enumerations

This is a list of all enumerations used by the XMLSpy 2005 API. If your scripting environment does not support enumerations use the number-values instead.

4.5.1 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

4.5.2 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:

spyAuthenticDocumentBegin	= 0
spyAuthenticDocumentEnd	= 1
spyAuthenticRangeBegin	= 2
spyAuthenticRangeEnd	= 3

4.5.3 SPYAuthenticElementActions

Description
Actions that can be used with GetAllowedElements (superseded by `AuthenticRange.CanPerformActionWith`).

Possible values:

k_ActionInsertAt	= 0
k_ActionApply	= 1
k_ActionClearSurr	= 2
k_ActionAppend	= 3
k_ActionInsertBefore	= 4
k_ActionRemove	= 5

4.5.4 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange and AuthenticView objects.

Possible values:

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

4.5.5 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility.

Possible values:

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

4.5.6 SPYDatabaseKind

Description
Values to select different kinds of databases for import. See `DatabaseConnection.DatabaseKind` for its use.

Possible values:

spyDB_Access	= 0
spyDB_SQLServer	= 1
spyDB_Oracle	= 2
spyDB_Sybase	= 3
spyDB_MySQL	= 4
spyDB_DB2	= 5
spyDB_Other	= 6
spyDB_Unspecified	= 7

© 2005 Altova GmbH

4.5.7 SPYDialogAction

Description
Values to simulate different interactions on dialogs. See Dialogs for all dialogs available.

Possible values:

spyDialogOK	= 0	`// simulate click on OK button`
spyDialogCancel	= 1	`// simulate click on Cancel button`
spyDialogUserInput	= 2	`// show dialog and allow user interaction`

4.5.8 SPYDOMType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyDOMType_msxml4	= 0
spyDOMType_xerces	= 1

4.5.9 SPYDTDSchemaFormat

Description
Enumeration to identify the different schema formats.

Possible values:

spyDTD	= 0
spyW3C	= 1

4.5.10 SPYEncodingByteOrder

Description
Enumeration values to specify encoding byte ordering for text import and export.

Possible values:

spyNONE	= 0
spyLITTLE_ENDIAN	= 1
spyBIG_ENDIAN	= 2

4.5.11 SPYExportNamespace

Description
Enumeration type to configure handling of namespace identifiers during export.

Possible values:

spyNoNamespace	= 0
spyReplaceColonWithUnderscore	= 1

4.5.12 SPYFrequentElements

Description
Enumeration value to parameterize schema generation.

Possible values:

spyGlobalElements	= 0
spyGlobalComplexType	= 1

4.5.13 SPYKeyEvent

Description
Enumeration type to identify the different key events. These events correspond with the equally named windows messages.

Possible values:

spyKeyDown	= 0
spyKeyUp	= 1
spyKeyPressed	= 2

4.5.14 SPYLibType

Description
Enumeration values to parameterize generation of C++ code from schema definitions.

Possible values:

spyLibType_static	= 0
spyLibType_dll	= 1

4.5.15 SPYLoading

Description
Enumeration values to define loading behaviour of URL files.

Possible values:

spyUseCacheProxy	= 0
spyReload	= 1

4.5.16 SPYMouseEvent

Description
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather then directly comparing with them.

Examples

```
' to check for ctrl-leftbutton-down in VB
If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or
XMLSpyLib.spyCtrlKeyDownMask)) Then
       ' react on ctrl-leftbutton-down
End If
```


© 2005 Altova GmbH

```
' to check for double-click with any button in VBScript
If (((i_eMouseEvent And spyDoubleClickMask) <> 0) Then
       ' react on double-click
End If
```

Possible values:

spyNoButtonMask	= 0	
spyMouseMoveMask	= 1	
spyLeftButtonMask	= 2	
spyMiddleButtonMask	= 4	
spyRightButtonMask	= 8	
spyButtonUpMask	= 16	
spyButtonDownMask	= 32	
spyDoubleClickMask	= 64	
spyShiftKeyDownMask	= 128	
spyCtrlKeyDownMask	= 256	
spyLeftButtonDownMask	= 34	// spyLeftButtonMask | spyButtonDownMask
spyMiddleButtonDownMask	= 36	// spyMiddleButtonMask | spyButtonDownMask
spyRightButtonDownMask	= 40	// spyRightButtonMask | spyButtonDownMask
spyLeftButtonUpMask	= 18	// spyLeftButtonMask | spyButtonUpMask
spyMiddleButtonUpMask	= 20	// spyMiddleButtonMask | spyButtonUpMask
spyRightButtonUpMask	= 24	// spyRightButtonMask | spyButtonUpMask
spyLeftDoubleClickMask	= 66	// spyRightButtonMask | spyButtonUpMask
spyMiddleDoubleClickMask	= 68	// spyMiddleButtonMask | spyDoubleClickMask
spyRightDoubleClickMask	= 72	// spyRightButtonMask | spyDoubleClickMask

4.5.17 SPYNumberDateTimeFormat

Description
Enumeration value to configure database connections.

Possible values:

spySystemLocale	= 0
spySchemaCompatible	= 1

4.5.18 SPYProgrammingLanguage

Description
Enumeration values to select the programming language for code generation from schema definitions.

Possible values:

spyUndefinedLanguage	= -1
spyJava	= 0
spyCpp	= 1
spyCSharp	= 2

4.5.19 SPYProjectItemTypes

Description
Enumeration values to identify the different elements in project item lists. See SpyProjectItem.ItemType.

Possible values:

spyUnknownItem	= 0
spyFileItem	= 1
spyFolderItem	= 2
spyURLItem	= 3

4.5.20 SPYProjectType

Description
Enumeration values to parameterize generation of C# code from schema definitions.

Possible values:

spyVisualStudioProject	= 0
spyBorlandProject	= 1
spyMonoMakefile	= 2

4.5.21 SPYSchemaDefKind

Description
Enumeration type to select schema diagram types.

Possible values:

© 2005 Altova GmbH

spyKindElement	= 0
spyKindComplexType	= 1
spyKindSimpleType	= 2
spyKindGroup	= 3
spyKindModel	= 4
spyKindAny	= 5
spyKindAttr	= 6
spyKindAttrGroup	= 7
spyKindAttrAny	= 8
spyKindIdentityUnique	= 9
spyKindIdentityKey	= 10
spyKindIdentityKeyRef	= 11
spyKindIdentitySelector	= 12
spyKindIdentityField	= 13
spyKindNotation	= 14
spyKindInclude	= 15
spyKindImport	= 16
spyKindRedefine	= 17
spyKindFacet	= 18
spyKindSchema	= 19
spyKindCount	= 20

4.5.22 SPYSchemaDocumentationFormat

Description
Enumeration values to parameterize generation of schema documentation. These values are used in `SchemaDocumentationDialog.OutputFormat`.

Possible values:

spySchemaDoc_HTML	= 0
spySchemaDoc_MSWord	= 1

4.5.23 SPYTextDelimiters

Description
Enumeration values to specify text delimiters for text export.

Possible values:

spyTabulator	= 0
spySemicolon	= 1
spyComma	= 2
spySpace	= 3

4.5.24 SPYTextEnclosing

Description
Enumeration value to specify text enclosing characters for text import and export.

Possible values:

spyNoEnclosing	= 0
spySingleQuote	= 1
spyDoubleQuote	= 2

4.5.25 SPYTypeDetection

Description
Enumeration to select how type detection works during GenerateDTDOrSchema.

Possible values:

spyBestPossible	= 0
spyNumbersOnly	= 1
spyNoDetection	= 2

4.5.26 SPYURLTypes

Description
Enumeration to specify different URL types.

Possible values:

spyURLTypeAuto	= -1
spyURLTypeXML	= 0
spyURLTypeDTD	= 1

4.5.27 SPYViewModes

Description
Enumeration values that define the different view modes for XML documents. The mode *spyViewContent(4)* identifies the mode that was intermediately called DocEdit mode and is now called Authentic mode. The mode *spyViewWSDL* identifies a mode which is mapped to the schema view on the GUI but distinguished internally.

Possible values:

spyViewGrid	= 0	
spyViewText	= 1	
spyViewBrowser	= 2	
spyViewSchema	= 3	
spyViewContent	= 4	// obsolete
spyViewAuthentic	= 4	
spyViewWSDL	= 5	

© 2005 Altova GmbH

4.5.28 SPYVirtualKeyMask

Description
Enumeration type for the most frequently used key masks that identify the status of the virtual keys. Use these values as bitmasks rather then directly comparing with them. When necessary, you can create further masks by using the 'logical or' operator.

Examples

```
'  VBScript sample: check if ctrl-key is pressed
If ((i_nVirtualKeyStatus And spyCtrlKeyMask) <> 0)) Then
       ' ctrl-key is pressed
End If

'  VBScript sample: check if ONLY ctrl-key is pressed
If (i_nVirtualKeyStatus == spyCtrlKeyMask) Then
       ' exactly ctrl-key is pressed
End If
```

```
// JScript sample: check if any of the right virtual keys is pressed
if ((i_nVirtualKeyStatus & (spyRightShiftKeyMask | spyRightCtrlKeyMask |
spyRightAltKeyMask)) != 0)
{
       ; ' right virtual key is pressed
}
```

Possible values:

spyNoVirtualKeyMask	= 0	
spyLeftShiftKeyMask	= 1	
spyRightShiftKeyMask	= 2	
spyLeftCtrlKeyMask	= 4	
spyRightCtrlKeyMask	= 8	
spyLeftAltKeyMask	= 16	
spyRightAltKeyMask	= 32	
spyShiftKeyMask	= 3	// spyLeftShiftKeyMask | spyRightShiftKeyMask
spyCtrlKeyMask	= 12	// spyLeftCtrlKeyMask | spyRightCtrlKeyMask
spyAltKeyMask	= 48	// spyLeftAltKeyMask | spyRightAltKeyMask

4.5.29 SPYXMLDataKind

Description
The different types of XMLData elements available for XML documents.

Possible values:

spyXMLDataXMLDocStruct	= 0
spyXMLDataXMLEntityDocStruct	= 1
spyXMLDataDTDDocStruct	= 2
spyXMLDataXML	= 3
spyXMLDataElement	= 4
spyXMLDataAttr	= 5
spyXMLDataText	= 6
spyXMLDataCData	= 7
spyXMLDataComment	= 8
spyXMLDataP	= 9
spyXMLDataDefDoctype	= 10
spyXMLDataDefExternalID	= 11
spyXMLDataDefElement	= 12
spyXMLDataDefAttlist	= 13
spyXMLDataDefEntity	= 14
spyXMLDataDefNotation	= 15
spyXMLDataKindsCount	= 16

© 2005 Altova GmbH

5 Using the XMLSpy API with Java

The XMLSpy API has an interface built up of Java classes, each of which corresponds to an object in the XMLSpy API. Developers can use these Java classes to interact with the COM API. These classes are listed below and described in subsequent sections. For a description of the XMLSpy API objects themselves, see the XMLSpy API documentation. Bear in mind that some API features are only available in scripting environments; these have therefore not been ported to Java.

Java classes

```
SpyApplication
  SpyProject
    SpyProjectItems
      SpyProjectItem
  SpyDocuments
    SpyDoc
      SpyAuthenticView
        SpyAuthenticRange
      SpyDocEditView
        SpyDocEditSelection
      SpyGridView
      SpyXMLData
  SpyDialogs
    SpyCodeGeneratorDlg
    SpyFileSelectionDlg
    SpySchemaDocumentationDlg
  SpyDatabaseConnection
  SpyElementList
  SpyElementListItem
  SpyExportSettings
  SpyTextImportExportSettings
```

Implementation of COM properties in Java

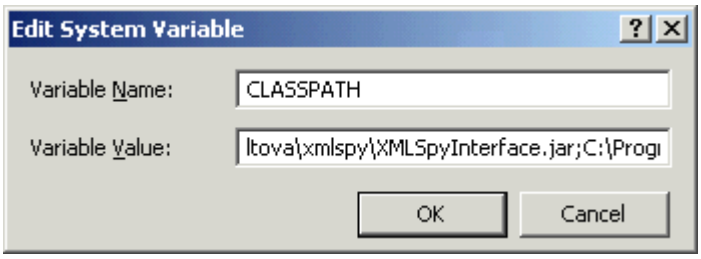
Properties in Java have been defined to include both a **`set`** and **`get`** method (`set` if it is allowed by the COM implementation). For example, the COM class `Document` contains the **`GridView`** property. In Java the method is called `SpyDoc` and the property is defined as a **`GetGridView`** method.

If you encounter compiling problems, please check the following points:

- The `xmlspylib.dll` must be available in `..\windows\system32`**.**
- The `XMLSpyInterface.jar` file must be inserted in the ClassPath environment variable.

Setting the `ClassPath` variable in Windows 2000 and XP

1. Click **Start | Settings | Control panel | System | Advanced | Environment Variables**. This opens the Environment Variables dialog box.
2. If a `ClassPath` entry already exists in the **System variables** group, select the `ClassPath` entry, and click the **Edit** button. Edit the path to: `"C:\Program Files\Altova\xmlspy\XMLSpyInterface.jar"`.



If a `ClassPath` entry does not exist in the System variables group, click the **New** button. The New System Variable dialog pops up. Enter `CLASSPATH` as the variable name, and `"C:\Program Files\Altova\xmlspy\XMLSpyInterface.jar"` as the `ClassPath` variable (alter the path to match your installation, if necessary).

© 2005 Altova GmbH

5.1 Sample source code

The "`SpyDoc doc = app.GetDocuments().OpenFile(...)`" command parameter must be altered to suit your environment.

What the sample does:

- Starts a new XMLSPY instance
- Opens the Datasheet.xml file (alter the path here...)
- Switches to the Enhanced Grid view
- Appends a new child element called "NewChild" with the text value "NewValuE" element to the root element
- Checks if the document is valid and outputs a message to the Java console
- Quits and releases the XMLSPY application

```
import XMLSpyInterface.*;

public class TestSpyInterface
{
 public TestSpyInterface() {}

 public static void main(String[] args)
 {
  SpyApplication app = new SpyApplication();
  app.ShowApplication( true );

  SpyDoc doc = app.GetDocuments().OpenFile("C:\\Program
Files\\Altova\\xmlspy2005\\Examples\\OrgChart.xml", true );

  doc.SwitchViewMode(SPYViewModes.spyViewGrid);
  SpyXMLData oData = doc.GetRootElement();
  SpyXMLData oNewChild = doc.CreateChild(SPYXMLDataKind.spyXMLDataElement);

  oNewChild.SetName( "NewChild" );
  oNewChild.SetTextValue("newVaLuE");
  oData.AppendChild(oNewChild);
  oNewChild.ReleaseInstance();
  oData.ReleaseInstance();


  if ( doc.IsValid(  ) == false )
  {
   // is to be expected after above insertion
   System.out.println( "!!!!!!validation error: "+doc.GetErrorString() );
   System.out.println( "!!!!!!validation error: "+doc.GetErrorPos() );
   System.out.println( "!!!!!!validation error: "+doc.GetBadData() );
  }

  doc.ReleaseInstance();
  app.Quit();
  app.ReleaseInstance();
 }
}
```

If you have difficulties compiling this sample, please try the following commands on the (**Start | Run | cmd**) command line. Please make sure you are currently in the folder that contains the sample java file.

compilation
javac -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface.java*

Execution
java -classpath c:*yourpathhere*\XMLSpyInterface.jar *testspyinterface*

© 2005 Altova GmbH

5.2 SpyApplication

```
public class SpyApplication
{
  public void ReleaseInstance();
  public void ShowApplication( boolean bShow );
  public void Quit ();
  public void AddMacroMenuItem( String sMacro, String sDisplayText );
  public void ClearMacroMenu( );
  public SpyDoc GetActiveDocument( );
  public SpyProject GetCurrentProject( );
  public SpyDocuments GetDocuments( );
  public SpyElementList GetDatabaseImportElementList( SpyDatabaseConnection
  oImportSettings );
  public SpyDatabaseConnection GetDatabaseSettings();
  public SpyElementList GetDatabaseTables( SpyDatabaseConnection
  oImportSettings );
  public SpyExportSettings GetExportSettings();
  public SpyElementList GetTextImportElementList( SpyTextImportExportSettings
  oImportSettings );
  public SpyTextImportExportSettings GetTextImportExportSettings();
  public SpyDoc ImportFromDatabase( SpyDatabaseConnection oImportSettings,
  SpyElementList oElementList );
  public SpyDoc ImportFromSchema( SpyDatabaseConnection oImportSettings,
  String strTable, SpyDoc oSchemaDoc );
  public SpyDoc ImportFromText( SpyTextImportExportSettings oImportSettings,
  SpyElementList oElementList );
  public SpyDoc ImportFromWord( String sFile );
  public void NewProject( String sPath, boolean bDiscardCurrent );
  public void OpenProject(String sPath , boolean bDiscardCurrent, boolean
  bDialog );
  public long ShowForm( String sName );
  public void URLDelete( String sURL, String sUser, String sPassword );
  public void URLMakeDirectory( String sURL, String sUser, String sPassword );
  public int GetWarningNumber();
  public String GetWarningText();

  // since Version 2004R4
  public SpyApplication GetApplication()
  public SpyApplication GetParent()
  public SpyDialogs GetDialogs()
  public boolean GetVisible()
  public void SetVisible( boolean i_bVisibility)
  public long GetWindowHandle()
}
```

5.3 SpyCodeGeneratorDlg

```
// since version 2004R4
public class SpyCodeGeneratorDlg
{
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public long GetProgrammingLanguage();
  public void SetProgrammingLanguage( long i_eVal );
  public String GetTemplateFileName();
  public void SetTemplateFileName( String i_strVal );
  public String GetOutputPath();
  public void SetOutputPath( String i_strVal );
  public long GetOutputPathDialogAction();
  public void SetOutputPathDialogAction( long i_eVal );
  public long GetPropertySheetDialogAction();
  public void SetPropertySheetDialogAction( long i_eVal );
  public long GetOutputResultDialogAction();
  public void SetOutputResultDialogAction( long i_eVal );
  public long GetCPPSettings_DOMType();
  public void SetCPPSettings_DOMType( long i_eVal );
  public long GetCPPSettings_LibraryType();
  public void SetCPPSettings_LibraryType( long i_eVal );
  public boolean GetCPPSettings_UseMFC();
  public void SetCPPSettings_UseMFC( boolean i_bVal );
  public long GetCSharpSettings_ProjectType();
  public void SetCSharpSettings_ProjectType( long i_eVal );
}
```


© 2005 Altova GmbH

5.4 SpyDatabaseConnection

```
public class SpyDatabaseConnection
{
  public void ReleaseInstance();
  public String GetADOConnection ( );
  public void SetADOConnection ( String sValue );
  public boolean GetAsAttributes ( );
  public void SetAsAttributes( boolean bValue );
  public boolean GetCreateMissingTables ( );
  public void SetCreateMissingTables( boolean bValue );
  public boolean GetCreateNew ( );
  public void SetCreateNew( boolean bValue );
  public boolean GetExcludeKeys ( );
  public void SetExcludeKeys ( boolean bValue );
  public String GetFile ( );
  public void SetFile ( String sValue );
  public boolean GetIncludeEmptyElements( );
  public void SetIncludeEmptyElements( boolean bValue );
  public long GetNumberDateTimeFormat( );
  public void SetNumberDateTimeFormat ( long nValue );
  public String GetODBCConnection( );
  public void SetODBCConnection( String sValue );
  public String GetSQLSelect( );
  public void SetSQLSelect( String sValue );
  public long GetTextFieldLen ( );
  public void SetTextFieldLen( long nValue );

  // since version 2004R4
  public long GetDatabaseKind ( )
  public void SetDatabaseKind( long nValue )
}
```

5.5 SpyDialogs

```
// Since version 2004R4
public class SpyDialogs
{
  public SpyApplication GetApplication();
  public SpyApplication GetParent();
  public SpyCodeGeneratorDlg GetCodeGeneratorDlg ();
  public SpyFileSelectionDlg GetFileSelectionDlg ();
  public SpySchemaDocumentationDlg GetSchemaDocumentationDlg ();
}
```


© 2005 Altova GmbH

5.6 SpyDoc

```
public class SpyDoc
{
  public void ReleaseInstance();
  public void SetEncoding( String strEncoding );
  public void SetPathName( String strPath );
  public String GetPathName( );
  public String GetTitle( );
  public boolean IsModified( );
  public void Save( );
  public void Close( boolean bDiscardChanges );
  public void UpdateViews( );
  public long GetCurrentViewMode( );
  public boolean SwitchViewMode( long nMode );
  public SpyGridView GetGridView( );
  public void SetActiveDocument( );
  public void StartChanges( );
  public void EndChanges( );
  public void TransformXSL( );
  public void AssignDTD( String sDTDFile, boolean bDialog );
  public void AssignSchema( String sSchemaFile, boolean bDialog );
  public void AssignXSL( String sXSLFile, boolean bDialog );
  public void ConvertDTDOrSchema( long nFormat, long nFrequentElements );
  public SpyXMLData CreateChild( long nKind );
  public void CreateSchemaDiagram( long nKind, String sName, String sFile );
  public SpyDocEditView GetDocEditView();
  public void ExportToDatabase( SpyXMLData oFromChild, SpyExportSettings
  oExportSettings, SpyDatabaseConnection oDatabaseConnection );
  public void ExportToText( SpyXMLData oFromChild, SpyExportSettings
  oExportSettings, SpyTextImportExportSettings oTextSettings );
  public void GenerateDTDOrSchema( long nFormat, int nValuesList, long
  nDetection, long nFrequentElements );
  public SpyElementList GetExportElementList( SpyXMLData oFromChild,
  SpyExportSettings oExportSettings );
  public SpyXMLData GetRootElement();
  public String SaveInString( SpyXMLData oData, boolean bMarked );
  public void SaveToURL( String sUrl, String sUser, String sPassword );
  public String GetErrorString();
  public int GetErrorPos();
  public SpyXMLData GetBadData();
  public boolean IsValid();
  public boolean IsWellFormed( SpyXMLData oData, boolean bWithChildren );

  // Since version 2004R3
  public SpyAuthenticView GetAuthenticView()

  // Since version 2004R4
  public SpyApplication GetApplication();
  public SpyDocuments GetParent();
  public String GetFullName();
  public void SetFullName( String i_strName );
  public String GetName();
  public String GetPath();
  public boolean GetSaved();
  public void SaveAs( String i_strFileNameOrPath );
  public String GetEncoding();
  public SpyXMLData GetDataRoot();
  public void GenerateProgramCode( SpyCodeGeneratorDlg i_dlg );
  public void AssignXSLFO( String i_strFile, boolean i_bUseDialog );
  public void TransformXSLFO ();
```

```
    public void GenerateSchemaDocumentation ( SpySchemaDocumentationDlg i_dlg );
}
```


© 2005 Altova GmbH

5.7 SpyDocuments

```
public class SpyDocuments
{
  public void ReleaseInstance();
  public long Count();
  public SpyDoc GetItem( long nNo );
  public SpyDoc NewFile( String strFile, String strType );
  public SpyDoc NewFileFromText( String nSource, String strType );
  public SpyDoc OpenFile( String sPath, boolean bDialog );
  public SpyDoc OpenURL( String sUrl, long nURLType, long nLoading, String
  sUser, String sPassword );
  public SpyDoc OpenURLDialog(String sURL, long nURLType, long nLoading,
  String sUser, String sPassword )
}
```

5.8 SpyElementList

```
public class SpyElementList
{
  public void ReleaseInstance();
  public long GetCount();
  public SpyElementListItem  GetItem( long nIndex );
  public void RemoveElement( long nIndex );
}
```


© 2005 Altova GmbH

5.9 SpyElementListItem

```
public class SpyElementListItem
{
  public void ReleaseInstance();
  public long GetElementKind();
  public void SetElementKind( long nKind );
  public long GetFieldCount ();
  public String GetName();
  public long GetRecordCount();
}
```

5.10 SpyExportSettings

```
public class SpyExportSettings
{
  public void ReleaseInstance();
  public boolean GetCreateKeys();
  public void SetCreateKeys( boolean bValue );
  public SpyElementList GetElementList();
  public void SetElementList( SpyElementList obj );
  public boolean GetEntitiesToText ();
  public void SetEntitiesToText ( boolean bValue );
  public boolean GetExportAllElements ();
  public void SetExportAllElements ( boolean bValue );
  public boolean GetFromAttributes ();
  public void SetFromAttributes ( boolean bValue );
  public boolean GetFromSingleSubElements ();
  public void SetFromSingleSubElements ( boolean bValue );
  public boolean GetFromTextValues ();
  public void SetFromTextValues ( boolean bValue );
  public boolean GetIndependentPrimaryKey ();
  public void SetIndependentPrimaryKey ( boolean bValue );
  public long GetNamespace ();
  public void SetNamespace ( long nValue );
  public int GetSubLevelLimit ();
  public void SetSubLevelLimit ( int nValue );
}
```


© 2005 Altova GmbH

5.11 SpyFileSelectionDlg

```
// Since version 2004R4
public class SpyFileSelectionDlg
{
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();
  public String GetFullName();
  public void SetFullName( String i_strName );
  public long GetDialogAction();
  public void SetDialogAction( long i_eAction );
}
```

5.12 SpyGridView

```
public class SpyGridView
{
  public void ReleaseInstance();
  public SpyXMLData  GetCurrentFocus();
  public void Deselect( SpyXMLData oData );
  public boolean GetIsVisible();
  public void Select( SpyXMLData oData );
  public void SetFocus( SpyXMLData oData );
}
```


© 2005 Altova GmbH

5.13 SpyProject

```
public class SpyProject
{
  public void ReleaseInstance();
  public void CloseProject( boolean bDiscardChanges, boolean bCloseFiles,
  boolean bDialog );
  public String GetProjectFile();
  public void SetProjectFile( String sFile );
  public SpyProjectItems GetRootItems();
  public void SaveProject();
  public void SaveProjectAs( String sPath, boolean bDialog );
}
```

5.14 SpyProjectItem

```
public class SpyProjectItem
{
  public void ReleaseInstance();
  public SpyProjectItems GetChildItems ();
  public String GetFileExtensions();
  public void SetFileExtensions( String sExtensions );
  public long GetItemType();
  public String GetName();
  public SpyDoc Open();
  public SpyProjectItem GetParentItem();
  public String GetPath();
  public String GetValidateWith();
  public void SetValidateWith( String sVal );
  public String GetXMLForXSLTransformation();
  public void SetXMLForXSLTransformation( String sVal );
  public String GetXSLForXMLTransformation();
  public void SetXSLForXMLTransformation( String sVal );
  public String GetXSLTransformationFileExtension();
  public void SetXSLTransformationFileExtension( String sVal );
  public String GetXSLTransformationFolder();
  public void SetXSLTransformationFolder( String sVal );
}
```


© 2005 Altova GmbH

5.15 SpyProjectItems

```
public class SpyProjectItems
{
  public void ReleaseInstance();
  public void AddFile( String sPath );
  public void AddFolder( String sName );
  public void AddURL( String sURL, long nURLType, String sUser, String
  sPassword, boolean bSave );
  public long Count();
  public SpyProjectItem GetItem( long nNumber );
  public void RemoveItem( SpyProjectItem oItemToRemove );
}
```

5.16 SpySchemaDocumentationDlg

```
// Since version 2004R4
public class SpySchemaDocumenttaionDlg
{
  public SpyApplication GetApplication();
  public SpyDialogs GetParent();

  public String GetOutputFile();
  public void SetOutputFile( String i_strVal );
  public long GetOutputFormat();
  public void SetOutputFormat( long i_eVal );

  public boolean GetShowResult();
  public void SetShowResult( boolean i_bVal );
  public long GetOptionsDialogAction();
  public void SetOptionsDialogAction( long i_eVal );
  public long GetOutputFileDialogAction();
  public void SetOutputFileDialogAction( long i_eVal );
  public boolean GetShowProgressBar();
  public void SetShowProgressBar( boolean i_bVal );

  public void IncludeAll( boolean i_bInclude );
  public boolean GetIncludeIndex();
  public void SetIncludeIndex( boolean i_bVal );
  public boolean GetIncludeGlobalElements();
  public void SetIncludeGlobalElements( boolean i_bVal );
  public boolean GetIncludeLocalElements();
  public void SetIncludeLocalElements( boolean i_bVal );
  public boolean GetIncludeGroups();
  public void SetIncludeGroups( boolean i_bVal );
  public boolean GetIncludeComplexTypes();
  public void SetIncludeComplexTypes( boolean i_bVal );
  public boolean GetIncludeSimpleTypes();
  public void SetIncludeSimpleTypes( boolean i_bVal );
  public boolean GetIncludeAttributeGroups();
  public void SetIncludeAttributeGroups( boolean i_bVal );
  public boolean GetIncludeRedefines();
  public void SetIncludeRedefines( boolean i_bVal );

  public void AllDetails( boolean i_bDetailsOn );
  public boolean GetShowDiagram();
  public void SetShowDiagram( boolean i_bVal );
  public boolean GetShowNamespace();
  public void SetShowNamespace( boolean i_bVal );
  public boolean GetShowType();
  public void SetShowType( boolean i_bVal );
  public boolean GetShowChildren();
  public void SetShowChildren( boolean i_bVal );
  public boolean GetShowUsedBy();
  public void SetShowUsedBy( boolean i_bVal );
  public boolean GetShowProperties();
  public void SetShowProperties( boolean i_bVal );
  public boolean GetShowSingleFacets();
  public void SetShowSingleFacets( boolean i_bVal );
  public boolean GetShowPatterns();
  public void SetShowPatterns( boolean i_bVal );
  public boolean GetShowEnumerations();
  public void SetShowEnumerations( boolean i_bVal );
  public boolean GetShowAttributes();
  public void SetShowAttributes( boolean i_bVal );
```

 © 2005 Altova GmbH

```
    public boolean GetShowIdentityConstraints();
    public void SetShowIdentityConstraints( boolean i_bVal );
    public boolean GetShowAnnotations();
    public void SetShowAnnotations( boolean i_bVal );
    public boolean GetShowSourceCode();
    public void SetShowSourceCode( boolean i_bVal );
}
```

5.17 SpyTextImportExportSettings

```
public class SpyTextImportExportSettings
{
  public void ReleaseInstance();
  public String GetDestinationFolder ();
  public void SetDestinationFolder ( String sVal );
  public long GetEnclosingCharacter ();
  public void SetEnclosingCharacter ( long nEnclosing );
  public String GetEncoding ();
  public void SetEncoding ( String sVal );
  public long GetEncodingByteOrder();
  public void SetEncodingByteOrder( long nByteOrder );
  public long GetFieldDelimiter();
  public void SetFieldDelimiter( long nDelimiter );
  public String GetFileExtension ();
  public void SetFileExtension ( String sVal );
  public boolean GetHeaderRow();
  public void SetHeaderRow( boolean bVal );
  public String GetImportFile();
  public void SetImportFile( String sVal );
}
```


© 2005 Altova GmbH

5.18 SpyXMLData

```
public class SpyXMLData
{
  public void ReleaseInstance();
  public void AppendChild( SpyXMLData oNewData );
  public void EraseAllChildren();
  public void EraseCurrentChild();
  public SpyXMLData GetCurrentChild();
  public SpyXMLData GetFirstChild( long nKind );
  public SpyXMLData GetNextChild();
  public boolean GetHasChildren();
  public void InsertChild( SpyXMLData oNewData );
  public boolean IsSameNode( SpyXMLData oToComp);
  public long GetKind();
  public boolean GetMayHaveChildren ();
  public String GetName();
  public void SetName( String sValue );
  public SpyXMLData GetParent();
  public String GetTextValue();
  public void SetTextValue ( String sValue );
}
```

5.19 Authentic

5.19.1 SpyAuthenticRange

```
// Since version 2004R3
public class SpyAuthenticRange
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyAuthenticView GetParent();
    public SpyAuthenticRange GotoNext(long eKind);
    public SpyAuthenticRange GotoPrevious(long eKind);
    public void Select();
    public long GetFirstTextPosition();
    public void SetFirstTextPosition (long nTextPosition);
    public long GetLastTextPosition();
    public void SetLastTextPosition (long nTextPosition);
    public String GetText();
    public void SetText (String strText);
    public boolean PerformAction(long eAction, String strElementName);
    public boolean CanPerformAction(long eAction, String strElementName);
    public String[] CanPerformActionWith (long eAction);
    public SpyAuthenticRange GoTo(long eKind, long nCount, long nFrom);
    public SpyAuthenticRange SelectNext (long eKind);
    public SpyAuthenticRange SelectPrevious (long eKind);
    public SpyAuthenticRange MoveBegin (long eKind, long nCount);
    public SpyAuthenticRange MoveEnd (long eKind, long nCount);
    public SpyAuthenticRange ExpandTo (long eKind);
    public SpyAuthenticRange CollapsToBegin ();
    public SpyAuthenticRange CollapsToEnd ();
    public SpyAuthenticRange GotoNextCursorPosition ();
    public SpyAuthenticRange GotoPreviousCursorPosition ();
    public boolean IsEmpty ();
    public boolean IsEqual (SpyAuthenticRange ipCmp);
    public SpyAuthenticRange Clone ();
    public SpyAuthenticRange SetFromRange (SpyAuthenticRange ipSrc);
    public boolean Delete ();
    public boolean Cut ();
    public boolean Copy ();
    public boolean Paste ();
    public SpyXMLData GetFirstXMLData ();
    public void SetFirstXMLData (SpyXMLData objXMLDataPtr);
    public long GetFirstXMLDataOffset ();
    public void SetFirstXMLDataOffset (long nOffset);
    public SpyXMLData GetLastXMLData ();
    public void SetLastXMLData (SpyXMLData objXMLDataPtr);
    public long GetLastXMLDataOffset ();
    public void SetLastXMLDataOffset (long nOffset);
    public String[] GetElementHierarchy ();
    public String[] GetElementAttributeNames (String strElementName);
    public boolean HasElementAttribute (String strElementName, String
strAttributeName);
    public String GetElementAttributeValue (String strElementName, String
strAttributeName);
    public void SetElementAttributeValue (String strElementName, String
strAttributeName, String strNewValue);
    public String[] GetEntityNames ();
    public void InsertEntity (String strEntityName);
    public boolean IsInDynamicTable ();
    public boolean AppendRow ();
    public boolean InsertRow ();
```

 © 2005 Altova GmbH

```
    public boolean DuplicateRow ();
    public boolean DeleteRow ();
    public boolean MoveRowUp ();
    public boolean MoveRowDown ();

    // Since version 2004R4
    public boolean IsCopyEnabled();
    public boolean IsCutEnabled();
    public boolean IsPasteEnabled();
    public boolean IsDeleteEnabled();
    public boolean IsTextStateApplied(String i_strElementName);
    public boolean IsFirstRow();
    public boolean IsLastRow();
}
```

5.19.2 SpyAuthenticView

```
// Since version 2004R3
public class SpyAuthenticView
{
    public void ReleaseInstance();
    public SpyApplication GetApplication();
    public SpyDoc GetParent();
    public SpyAuthenticRange GetSelection();
    public void SetSelection (SpyAuthenticRange obj);
    public SpyAuthenticRange GetDocumentBegin();
    public SpyAuthenticRange GetDocumentEnd();
    public SpyAuthenticRange GetWholeDocument();
    public long GetMarkupVisibility();
    public void SetMarkupVisibility (long eSpyAuthenticMarkupVisibility);
    public SpyAuthenticRange GoTo(long eKind, long nCount, long nFrom);
    public void Print(boolean bWithPreview, boolean bPromptUser);
    public boolean Undo();
    public boolean Redo();
    public void UpdateXMLInstanceEntities();

    // Since version 2004R4
    public String GetAsXMLString();
    public void SetAsXMLString(String i_strXML);
    public SpyXMLData GetXMLDataRoot();
    public boolean IsUndoEnabled ();
    public boolean IsRedoEnabled ();
}
```

5.19.3 SpyDocEditSelection

```
public class SpyDocEditSelection
{
  public void ReleaseInstance();
  public SpyXMLData GetEnd();
  public long GetEndTextPosition();
  public SpyXMLData GetStart();
  public long GetStartTextPosition();
}
```

5.19.4 SpyDocEditView

```
public class SpyDocEditView
{
```

```
  public void ReleaseInstance();
  public void ApplyTextState( String sElementName );
  public SpyDocEditSelection GetCurrentSelection();
  public void EditClear();
  public void EditCopy();
  public void EditCut();
  public void EditPaste();
  public void EditRedo();
  public void EditSelectAll();
  public void EditUndo();
  public SpyXMLData GetNextVisible( SpyXMLData oElement );
  public SpyXMLData GetPreviousVisible( SpyXMLData oElement );
  public boolean GetIsEditClearEnabled ();
  public boolean GetIsEditCopyEnabled ();
  public boolean GetIsEditCutEnabled ();
  public boolean GetIsEditPasteEnabled ();
  public boolean GetIsEditRedoEnabled ();
  public boolean GetIsEditUndoEnabled ();
  public boolean GetIsRowAppendEnabled ();
  public boolean GetIsRowDeleteEnabled ();
  public boolean GetIsRowDuplicateEnabled();
  public boolean GetIsRowInsertEnabled();
  public boolean GetIsRowMoveDownEnabled();
  public boolean GetIsRowMoveUpEnabled();
  public boolean IsTextStateApplied( String sElementName );
  public boolean IsTextStateEnabled( String sElementName );
  public void LoadXML( String sXML );
  public void MarkUpView( long nKind );
  public void RowAppend();
  public void RowDelete();
  public void RowDuplicate();
  public void RowInsert();
  public void RowMoveDown();
  public void RowMoveUp();
  public String SaveXML();
  public void SelectionMoveTabOrder( boolean bForward, boolean bTag );
  public boolean SelectionSet( SpyXMLData oStart, long nStartPos,SpyXMLData
  oEndElement, long nEndPos );
  public SpyXMLData GetXMLRoot();
  public String[] GetAllowedElements( long nAction, SpyXMLData oStartPtr,
  SpyXMLData oEndPtr );
}
```


© 2005 Altova GmbH

5.20 Predefined constants

This section lists all classes that define the predefined constants used by the Java interface.

5.20.1 SPYAuthenticActions

```
public class SPYAuthenticActions
{
    public final static long spyAuthenticInsertAt     = 0;
    public final static long spyAuthenticApply        = 1;
    public final static long spyAuthenticClearSurr    = 2;
    public final static long spyAuthenticAppend       = 3;
    public final static long spyAuthenticInsertBefore = 4;
    public final static long spyAuthenticRemove       = 5;
}
```

5.20.2 SPYAuthenticDocumentPosition

```
public class SPYAuthenticDocumentPosition
{
    public final static long                        = 0;
    spyAuthenticDocumentBegin
    public final static long spyAuthenticDocumentEnd = 1;
    public final static long spyAuthenticRangeBegin  = 2;
    public final static long spyAuthenticRangeEnd    = 3;
}
```

5.20.3 SPYAuthenticElementKind

```
public class SPYAuthenticElementKind
{
    public final static long spyAuthenticChar      = 0;
    public final static long spyAuthenticWord      = 1;
    public final static long spyAuthenticLine      = 3;
    public final static long spyAuthenticParagraph = 4;
    public final static long spyAuthenticTag       = 6;
    public final static long spyAuthenticDocument  = 8;
    public final static long spyAuthenticTable     = 9;
    public final static long spyAuthenticTableRow  = 10;
    public final static long                       = 11;
    spyAuthenticTableColumn
}
```

5.20.4 SPYAuthenticMarkupVisibility

```
public class SPYAuthenticMarkupVisibility
{
    public final static long spyAuthenticMarkupHidden = 0;
    public final static long spyAuthenticMarkupSmall  = 1;
    public final static long spyAuthenticMarkupLarge  = 2;
    public final static long spyAuthenticMarkupMixed  = 3;
}
```

5.20.5 SPYDatabaseKind

```
public class SPYLoading
{
    public final static long spyDB_Access       = 0;
    public final static long spyDB_SQLServer    = 1;
    public final static long spyDB_Oracle       = 2;
    public final static long spyDB_Sybase       = 3;
    public final static long spyDB_MySQL        = 4;
    public final static long spyDB_DB2          = 5;
    public final static long spyDB_Other        = 6;
    public final static long spyDB_Unspecified  = 7;
}
```

5.20.6 SPYDialogAction

```
public class SPYDialogAction
{
    public final static long spyDialogOK        = 0;
    public final static long spyDialogCancel    = 1;
    public final static long spyDialogUserInput = 2;
}
```

5.20.7 SPYDOMType

```
public class SPYDOMType
{
    public final static long spyDOMType_msxml4 = 0;
    public final static long spyDOMType_xerces = 1;
}
```


© 2005 Altova GmbH

5.20.8 SPYDTDSchemaFormat

```
public class SPYDTDSchemaFormat
{
    public final static long spyDTD     = 0;
    public final static long spyDCD     = 1;
    public final static long spyXMLData = 2;
    public final static long spyBizTalk = 3;
    public final static long spyW3C     = 4;
}
```

5.20.9 SPYEncodingByteOrder

```
public class SPYEncodingByteOrder
{
    public final static long spyNONE          = 0;
    public final static long spyLITTLE_ENDIAN = 1;
    public final static long spyBIG_ENDIAN    = 2;
}
```

5.20.10 SPYExportNamespace

```
public class SPYExportNamespace
{
    public final static long spyNoNamespace                = 0;
    public final static long spyReplaceColonWithUnderscore = 1;
}
```

5.20.11 SPYFrequentElements

```
public class SPYFrequentElements
{
    public final static long spyGlobalElements    = 0;
    public final static long spyGlobalComplexType = 1;
}
```

5.20.12 SPYLibType

```
public class SPYLibType
{
    public final static long spyLibType_static = 0;
    public final static long spyLibType_dll    = 1;
}
```

5.20.13 SPYLoading

```
public class SPYLoading
{
    public final static long spyUseCacheProxy = 0;
    public final static long spyReload        = 1;
}
```

5.20.14 SPYNameDateTimeFormat

```
public class  SPYNumberDateTimeFormat
{
    public final static long spySystemLocale     = 0;
    public final static long spySchemaCompatible = 1;
}
```

5.20.15 SPYProgrammingLanguage

```
public class SPYLoading
{
    public final static long spyUndefinedLanguage = -1;
    public final static long spyJava               = 0;
    public final static long spyCpp                = 1;
    public final static long spyCSharp             = 2;
}
```

5.20.16 SPYProjectItemTypes

```
public class SPYProjectItemTypes
{
    public final static long spyUnknownItem = 0;
    public final static long spyFileItem    = 1;
    public final static long spyFolderItem  = 2;
    public final static long spyURLItem     = 3;
}
```

5.20.17 SPYProjectType

```
public class SPYProjectType
{
    public final static long spyVisualStudioProject     = 0;
    public final static long spyVisualStudio2003Project = 1;
    public final static long spyBorlandProject          = 2;
    public final static long spyMonoMakefile            = 3;
}
```


© 2005 Altova GmbH

5.20.18 SPYSchemaDefKind

```
public class SPYSchemaDefKind
{
    public final static long spyKindElement          = 0;
    public final static long spyKindComplexType      = 1;
    public final static long spyKindSimpleType       = 2;
    public final static long spyKindGroup            = 3;
    public final static long spyKindModel            = 4;
    public final static long spyKindAny              = 5;
    public final static long spyKindAttr             = 6;
    public final static long spyKindAttrGroup        = 7;
    public final static long spyKindAttrAny          = 8;
    public final static long spyKindIdentityUnique   = 9;
    public final static long spyKindIdentityKey      = 10;
    public final static long spyKindIdentityKeyRef   = 11;
    public final static long spyKindIdentitySelector = 12;
    public final static long spyKindIdentityField    = 13;
    public final static long spyKindNotation         = 14;
    public final static long spyKindInclude          = 15;
    public final static long spyKindImport           = 16;
    public final static long spyKindRedefine         = 17;
    public final static long spyKindFacet            = 18;
    public final static long spyKindSchema           = 19;
    public final static long spyKindCount            = 20;
}
```

5.20.19 SPYSchemaDocumentationFormat

```
public class SPYSchemaDocumentationFormat
{
    public final static long spySchemaDoc_HTML    = 0;
    public final static long spySchemaDoc_MSWord  = 1;
}
```

5.20.20 SPYTextDelimiters

```
public class SPYTextDelimiters
{
```

```
    public final static long spyTabulator  = 0;
    public final static long spySemicolon  = 1;
    public final static long spyComma      = 2;
    public final static long spySpace      = 3;
}
```

5.20.21 SPYTextEnclosing

```
public class SPYTextEnclosing
{
    public final static long spyNoEnclosing = 0;
    public final static long spySingleQuote = 1;
    public final static long spyDoubleQuote = 2;
}
```

5.20.22 SPYTypeDetection

```
public class SPYTypeDetection
{
    public final static long spyBestPossible = 0;
    public final static long spyNumbersOnly  = 1;
    public final static long spyNoDetection  = 2;
}
```

5.20.23 SPYURLTypes

```
public class SPYURLTypes
{
    public final static long spyURLTypeAuto = (-1);
    public final static long spyURLTypeXML  = 0;
    public final static long spyURLTypeDTD  = 1;
}
```

5.20.24 SpyViewModes

```
public class SPYViewModes
{
    public final static long spyViewGrid    = 0;
    public final static long spyViewText    = 1;
    public final static long spyViewBrowser = 2;
    public final static long spyViewSchema  = 3;
    public final static long spyViewContent = 4;
}
```


© 2005 Altova GmbH

5.20.25 SPYWhitespaceComparison

```
public class SPYWhitespaceComparison
{
    public final static long spyCompareAsIs       = 0;
    public final static long spyCompareNormalized = 1;
    public final static long spyStripAll          = 2;
}
```

5.20.26 SPYXMLDataKind

```
public class SPYXMLDataKind
{
    public final static long spyXMLDataXMLDocStruct       = 0;
    public final static long spyXMLDataXMLEntityDocStruct = 1;
    public final static long spyXMLDataDTDDocStruct       = 2;
    public final static long spyXMLDataXML                = 3;
    public final static long spyXMLDataElement            = 4;
    public final static long spyXMLDataAttr               = 5;
    public final static long spyXMLDataText               = 6;
    public final static long spyXMLDataCData              = 7;
    public final static long spyXMLDataComment            = 8;
    public final static long spyXMLDataPI                 = 9;
    public final static long spyXMLDataDefDoctype         = 10;
    public final static long spyXMLDataDefExternalID      = 11;
    public final static long spyXMLDataDefElement         = 12;
    public final static long spyXMLDataDefAttlist         = 13;
    public final static long spyXMLDataDefEntity          = 14;
    public final static long spyXMLDataDefNotation        = 15;
    public final static long spyXMLDataKindsCount         = 16;
}
```

Altova XMLSpy 2005 Enterprise Edition

Anhänge

Anhänge

Die Anhänge enthalten technische Hintergrundinformationen sowie Informationen zu Lizenz und Vertrieb Ihres Produkts. In diesem Abschnitt finden Sie Informationen zu den folgenden Themen:

Technische Hintergrundinformationen

- XML Parser
- Schema-Dialekte
- XSLT-Prozessor
- OS- und Arbeitsspeicheranforderungen
- Internet-Verwendung
- Unicode-Unterstützung
- Lizenzüberwachung

Lizenzierung und Vertrieb

- Vertrieb durch "Electronic Software Distribution"
- Copyright
- Software-Produktlizenz

© 2005 Altova GmbH

1 Technische Hintergrundinformationen

Dieses Kapitel enthält nützliche Hintergrundinformationen zu den technischen Aspekten von XMLSpy 2005. Wenn Ihnen in einem anderen Teil dieses Hilfesystems ein technischer Begriff unterkommt, finden Sie an manchen Stellen einen Link zu einigen der Hintergrundinformationen in diesem Abschnitt.

1.1 XML Parser

Beim Öffnen eines XML-Dokuments verwendet XMLSpy 2005 seinen integrierten Validierungsparser, um das Dokument auf Wohlgeformtheit zu überprüfen und es gegen ein beliebiges DTD-, DCD-, BizTalk- oder XSD-Schema zu validieren.

Derselbe Parser kommt auch beim Editieren eines Dokuments zum Einsatz, das sich auf eine DTD, DCD, BizTalk oder XSD-Schema bezieht. Er bietet intelligente Editierhilfen und zeigt jeden Validierungsfehler sofort an.

Dies wird durch das Konzept der inkrementellen Validierung des neuen Parsers ermöglicht, der für die speziellen Anforderungen einer IDE optimiert wurde.

Der integrierte Parser implementiert die Final Recommendation des W3C für XML-Schema. Wir verfolgen die Arbeit der W3C Schema Group ständig und nehmen aktiv an allen schemaspezifischen Diskussionen teil, um in der Lage zu sein, Ihnen eine Entwicklungsumgebung auf dem neusten Stand der Technik zu bieten.

© 2005 Altova GmbH

1.2 Schema-Dialekte

Ein wichtiger Aspekt von XML sind Schemas und DTDs, die die logische Struktur (oder das Content Model) eines XML-Dokuments definieren. XMLSpy 2005 ist das ideale Tool, in dem die Erstellung von DTDs und Schemas integriert ist, während Sie mit XML-Instanzdokumenten arbeiten.

XMLSpy 2005 unterstützt sowohl das Bearbeiten als auch die Validierung von Schemas der folgenden Arten:

- Document Type Definitions (DTD)
- Document Content Descriptions (DCD)
- BizTalk
- XML Schema Definition (XSD) Draft 7. April, 2000, CR 24. Okt. 2000, 2. Mai 2001 Final Recommendation

und kann ein XML-Instanzdokument gegen jeden der oben angeführten Schema-Dialekte validieren.

1.3 Altova XSLT- und XQuery-Prozessor

Die Altova Produkte arbeiten mit dem Altova XSLT 1.0-Prozessor, dem Altova XSLT 2.0-Prozessor und dem Altova XQuery-Prozessor. Nähere Informationen zu diesen Prozessoren finden Sie im Abschnitt "Benutzerreferenz" dieser Dokumentation.

Diese drei Prozessoren stehen auch als eigenständige Applikationen zum kostenlosen Download von der Altova Website zur Verfügung. Die Dokumentation zur Verwendung der Prozessoren wird mit dem jeweiligen Applikationspaket mitgeliefert.

© 2005 Altova GmbH

1.4 OS- und Arbeitsspeicheranforderungen

XMLSpy 2005 ist eine moderne 32-Bit Windows-Applikation, die auf Windows NT 4.0, Windows 2000 und Windows XP läuft. Sie benötigen dafür einigen Arbeitsspeicher auf Ihrem System, da jedes Dokument zur Gänze in den Arbeitsspeicher geladen wird.

Andererseits wird dafür normalerweise weniger Arbeitsspeicher als für viele Java-Applikationen benötigt, da das Programm ausschließlich in C++ geschrieben wurde und daher nicht so viel Platz benötig wie eine Java Runtime-Umgebung.

Es ist notwendig, dass die Dokumente in den Hauptarbeitsspeicher geladen werden, damit jedes Dokument zur Gänze geparst und analysiert werden kann und die Anzeige- und Bearbeitungsgeschwindigkeit während der normalen Arbeit verbessert wird. Während für die Bearbeitung eines kleinen bis mittleren Dokuments (bis zu 512 KB) nur 2 MB RAM ausreichen, beansprucht ein 5 MB-Dokument während des ersten Parsens bis zu 50 MB.

Auch die unbegrenzte Rückgängig-Funktion kann einiges an Arbeitsspeicher in Anspruch nehmen. Wenn Sie große Abschnitte in großen Dokumenten immer wieder ausschneiden und einfügen, kann dies enorm viel Speicherplatz verbrauchen.

1.5 Internet-Verwendung

XMLSpy 2005 ist eine IDE (Integrated Development Environment) für XML und startet daher für Sie auch Internet-Verbindungen. Dies geschieht in den folgenden Fällen:

- Wenn Sie im Registrierungsdialogfeld auf "Kostenlosen Evaluierungs-Key anfordern" klicken, wird über eine reguläre HTTP (Port 80)-Verbindung eine Verbindung mit unserem Webserver hergestellt. Sie erhalten dann per E-Mail (normales SMTP) den kostenlosen Evaluierungs-Keycode zugesandt.
- Falls Sie das Dialogfeld URL öffnen verwenden, um ein Dokument direkt über eine URL zu öffnen, wird dieses Dokument über eine HTTP-Verbindung (Port 80) aufgerufen.
- Wenn Sie ein XML-Dokument öffnen, das sich auf ein XML-Schema oder eine DTD bezieht und das Dokument durch eine URL definiert wird, wird es ebenfalls über eine HTTP-Verbindung (Port 80) aufgerufen, sobald Sie das XML-Dokument validieren. Dies kann auch automatisch beim Öffnen eines Dokuments vorkommen, wenn Sie XMLSpy 2005 so konfiguriert haben, dass Dateien beim Öffnen automatisch validiert werden Register "Datei" des Dialogfelds Extras | Optionen.
- Wenn Sie den Befehl Als Mail senden... verwenden, wird die aktuelle Auswahl bzw. Datei über ein MAPI-kompatibles Mail-Programm, das auf dem PC des Benutzers installiert ist, versendet.

Alle diese Verbindungen werden natürlich nur als Antwort auf eine direkte Anforderung von Ihnen initiiert! Schließlich wird XML auch für die Arbeit mit dem Internet verwendet, daher muss jedes XML-Entwicklungstool über Internet-Zugriffsprotokolle verfügen, um dem XML-Entwickler eine effiziente Umgebung für seine täglichen Aufgaben zu bieten.

© 2005 Altova GmbH

1.6 Unicode-Unterstützung

Unicode ist ein neuer 16-Bit-Zeichensatzstandard, der vom Unicode-Konsortium definiert wurde und der eine eindeutige Zahl für jedes Zeichen zur Verfügung stellt, und zwar:

- plattformunabhängig
- programmunabhängig
- unabhängig von der Sprache

Im Prinzip arbeiten Computer nur mit Zahlen. Sie speichern Buchstaben und andere Zeichen unter einer Zahl, die sie diesen zuweisen. Vor der Entwicklung von Unicode gab es hunderte von verschiedenen Codiersystemen für die Zuweisung dieser Zahlen. Es gab keine einzige Codierung, die Platz für genügend Zeichen bot: So werden z.B. in der Europäischen Union mehrere verschiedene Codierungen benötigt, um alle ihre Sprachen abzudecken. Selbst für eine einzige Sprache wie z.B. Englisch reichte eine einzelne Codierung nicht aus, um alle gebräuchlichen Buchstaben, Satzzeichen und technischen Symbole abzudecken.

Diese Codierungssysteme standen früher miteinander in Konflikt, d.h. zwei Codierungen verwendeten dieselbe Zahl für zwei verschiedene Zeichen oder verschiedene Zahlen für dasselbe Zeichen. Jeder Computer (v.a. Server) muss viele verschiedene Codierungen unterstützen. Wenn aber Daten zwischen verschiedenen Codierungen oder Plattformen ausgetauscht werden, besteht immer das Risiko, dass Fehler entstehen.

Unicode ist die Lösung für diese Probleme!
Unicode enthält eine eindeutige Zahl für jedes Zeichen, egal für welche Plattform, welches Programm oder welche Sprache. Der Unicode-Standard wurde von branchenführenden Unternehmen wie Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Base und vielen anderen übernommen.

Unicode ist für moderne Standards wie XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, usw. erforderlich und ist das offizielle Verfahren zum Implementieren von ISO/IEC 10646. Unicode wird von vielen Betriebssystemen, allen modernen Browsern und vielen anderen Produkten unterstützt. Die Entwicklung des Unicode-Standards und die Verfügbarkeit von Tools, die diesen unterstützen, gehören zu den wichtigsten globalen Software Technologie-Trends in letzter Zeit.

Die Integration von Unicode in Client-Server oder Multi-Tier-Applikationen und Websites bietet beträchtliche Kostenvorteile im Vergleich zur Verwendung herkömmlicher Zeichensätze. Dank Unicode kann ein einziges Software-Produkt oder eine einzige Website für mehrere Plattformen, Sprachen und Länder konzipiert werden, ohne dass es neu entwickelt werden muss. Daten können dadurch durch viele verschiedene Systeme übertragen werden, ohne dass dabei Fehler auftreten.

XML basiert zwar definitionsgemäß auf dem Unicode-Standard, doch zählt XMLSpy 2005 bisher immer noch zu einem der wenigen XML-Entwicklungstools, die Unicode *vollständig* implementieren!

1.6.1 Windows NT4.0/2000/XP

Ab Version 2.0 bot XMLSpy 2005 vollständige Unicode-Unterstützung in den Software-Versionen für Windows NT, Windows 2000 und Windows XP. Um ein XML-Dokument von einem System mit nicht lateinischer Zeichensatzcodierung bearbeiten zu können, benötigen Sie allerdings auch eine Schriftart, die die von diesem Dokument verwendeten Unicode-Zeichen unterstützt.

Windows NT bietet bei seinen Schriftarten Arial, Times und Courier New normalerweise Unterstützung für alle gebräuchlichen Single-Byte-Schriftsysteme sowie des Weiteren alle in

Ihrem Land erforderlichen Schriftsysteme (d.h. wenn Sie die japanische Version von Windows NT installieren, haben Sie automatisch Schriftarten, die Katakana, Hiragana und Kanji unterstützen, sowie Eingabemethoden und Wörterbücher für die Eingabe von Kanji und zum Wechseln zwischen Katakana und Hiragana). Wenn Sie also Dokumente aus einem fremden Schriftsystem bearbeiten möchten, sollten Sie die zusätzlichen Windows NT-Komponenten für dieses Schriftsystem installieren oder spezielle Unicode-Schriftarten für diese Schriftsysteme zukaufen (erhältlich bei allen führenden Anbietern von Schriftarten).

Beachten Sie bitte, dass die meisten Schriftarten nur eine bestimmte Untergruppe des gesamten Unicode-Bereichs enthalten und normalerweise für das entsprechende Schriftsystem ausgelegt sind. Aus diesem Grund kann es vorkommen, dass ein XML-Dokument "nicht druckbare" Zeichen enthält, da die gewählte Schriftart die erforderlichen Glyphen nicht enthält. Es ist daher manchmal nützlich, wenn man eine Schriftart hat, die den gesamten Unicode-Bereich abdeckt - v.a. wenn Sie XML-Dokumente aus der ganzen Welt editieren.

Die größte uns bekannte Universalschriftart ist "Arial Unicode MS", eine Schriftart, die von Agfa Monotype für Microsoft entwickelt wurde. Diese Schriftart enthält mehr als 50.000 Glyphen und deckt das gesamte im Unicode 2.1.-Standard definierte Zeichenspektrum ab. Die Schriftart ist in Microsoft Office 2000 enthalten und beansprucht 23 MB.

Wir empfehlen Ihnen dringend, diese Schriftart auf Ihrem System zu installieren und mit XMLSpy 2005 zu verwenden, wenn Sie öfter Dokumente bearbeiten, die in unterschiedlichen Schriftsystemen verfasst wurden. Diese Schriftart wird bei Verwendung der Standardinstallation von Microsoft Office nicht installiert, Sie können jedoch die Option "Benutzerdefinierte Installation" wählen, um sie zu installieren.

Im Ordner "Examples" finden Sie außerdem eine neue XHTML-Datei mit dem Namen `Unicode-UTF8.html`, die den Satz "Wenn die Welt miteinander spricht, spricht sie Unicode" in vielen verschiedenen Sprachen und Schriftsystemen enthält ("When the world wants to talk, it speaks Unicode") (世界的に話すなら、Unicode です。) - Diese Zeile stammt aus der 10. Unicode-Konferenz von 1997 und ist ein schönes Beispiel für die Bedeutung von Unicode für den XML-Standard. Wenn Sie diese Datei öffnen, erhalten Sie einen kurzen Eindruck davon, was mit Unicode möglich ist und welche Schriftsysteme von den auf Ihrem PC verfügbaren Schriftarten unterstützt werden.

1.6.2 Rechts nach Links-Schreibrichtung

Beachten Sie bitte, dass selbst unter Windows NT 4.0 Texte aus Schriftsystemen mit einer Schreibrichtung von rechts nach links (z.B. Hebräisch oder Arabisch) nur in den Ländern korrekt wiedergegeben werden, in denen solche Schriftsysteme tatsächlich verwendet werden. Dies liegt daran, dass nur die hebräische und arabische Version von Windows NT die Wiedergabe und Bearbeitung von Text in Rechts-links-Schreibweise auf Betriebssystemebene unterstützen.

© 2005 Altova GmbH

1.7 Lizenzüberwachung

XMLSpy 2005 verfügt über ein integriertes Lizenzüberwachungsmodul, das Ihnen hilft, unbeabsichtigte Verletzungen der Lizenzvereinbarung zu vermeiden. XMLSpy 2005 kann entweder mit einer Einzelplatzlizenz oder einer Mehrfachlizenz erworben werden. Je nach Lizenz stellt das Lizenzüberwachungsmodul sicher, dass nicht mehr als die lizenzierte Anzahl an Benutzern XMLSpy 2005 gleichzeitig verwendet.

Die Kommunikation zwischen verschiedenen Instanzen von XMLSpy 2005, die auf verschiedenen Computern laufen, erfolgt über Ihr LAN.

Einzelplatzlizenz
Beim Start von XMLSpy 2005 wird ein kurzes Datagram abgesendet, um andere Instanzen des Produkts, die auf anderen Computern im selben Netzwerk laufen, zu finden. Wenn keine Antwort einlangt, wird ein Port geöffnet, der Informationen von anderen Instanzen von XMLSpy 2005 empfangen kann. Mehr geschieht nicht bei einer Einzelplatzlizenz. Wenn Sie nicht mit einem LAN verbunden sind oder für Ihre Internetverbindung eine Einwahlverbindung verwenden, verursacht XMLSpy 2005 keinerlei Netzwerkverkehr.

Mehrfachlizenz
Wenn Sie im selben LAN mehrere Kopien von XMLSpy 2005 verwenden, kommunizieren diese beim Start kurz miteinander, um Keycode-Informationen auszutauschen um sicherzustellen, dass nicht mehr als die lizenzierte Anzahl an Lizenzen gleichzeitig in Verwendung ist.

Dieselbe Lizenzüberwachungstechnologie wird auch bei Unix und vielen anderen Datenbankentwicklungstools verwendet. Sie gestattet Benutzern den Erwerb von Parallellizenzen für mehrere Benutzer zu vernünftigen Preisen. (unsere Preisliste finden Sie unter http://www.altova.com/order).

Beachten Sie, dass XMLSpy 2005 niemals versucht, Informationen aus Ihrem LAN oder über das Internet zu versenden. Wir sind außerdem bestrebt, nur wenige, kleine Netzwerkpakete zu versenden, um Ihr Netzwerk nicht zu überlasten. Die von XMLSpy 2005 verwendeten TCP/IP Ports (2799) sind offiziell bei IANA registriert, (nähere Informationen siehe http://www.isi.edu/in-notes/iana/assignments/port-numbers) und unser Lizenzüberwachungsmodul basiert auf einer bewährten und getesteten Technologie.

Wenn Sie eine Firewall verwenden, werden Sie unter Umständen feststellen, dass die Computer, auf denen XMLSpy 2005 läuft, über Port 2799 miteinander kommunizieren. Sie können diesen Netzwerkverkehr zwischen verschiedenen Gruppen in Ihrem Unternehmen natürlich blockieren, solange Sie mit anderen Mittel sicherstellen können, dass Ihre Lizenzvereinbarung eingehalten wird.

Sie werden außerdem auch feststellen, dass XMLSpy 2005 viele nützliche Funktionen enthält, die Ihre Internet-Verbindung verwenden, diese stehen jedoch nicht mit der Lizenzüberwachungsfunktion in Zusammenhang.

2 Lizenzierung und Vertrieb

Dieser Anhang enthält wichtige rechtliche Informationen zur Verwendung dieser Software. Lesen Sie die Informationen bitte sorgfältig - sie sind rechtlich bindend, da Sie sich bei der Installation dieses Software-Produkts damit einverstanden erklärt haben.

© 2005 Altova GmbH

2.1 Electronic Software Distribution (EDS)

XMLSpy 2005 kann nur über EDS (Electronic Software Distribution), also auf elektronischem Weg erworben werden, da diese Methode unseren Kunden unzählige Vorteile bietet:

- Sie können die Software kostenlos testen, bevor Sie sich zu einem Kauf entscheiden.
- Wenn Sie sich entschieden haben, die Software zu kaufen, können Sie Ihre Bestellung online abgeben und das Produkt steht Ihnen in Minutenschnelle im vollen Lizenzumfang zur Verfügung.
- Sie erhalten immer die neueste Version unserer Software
- Die Software enthält ein umfassendes Online-Hilfesystem und ein Handbuch in digitaler Form, das Sie jederzeit ausdrucken können, wenn Sie die Dokumentation lieber auf Papier lesen.

Nachdem Sie dieses Software-Produkt heruntergeladen haben, können Sie XMLSpy 2005 30 Tage lang kostenlos testen. Während dieses Zeitraums werden Sie nach etwa 20 Tagen in regelmäßigen Abständen daran erinnert, dass die Software noch nicht lizenziert wurde. Diese Erinnerungsmeldung wird allerdings nur einmal, nämlich bei jedem Start des Programms, angezeigt.

Wenn Sie das Programm nach Ablauf des 30-tägigen Evaluierungszeitraums weiterverwenden möchten, müssen Sie eine SOFTWARE-PRODUKTLIZENZ erwerben, die Sie in Form eines Keycodes erhalten, der in das Registrierungsdialogfeld eingegeben werden muss, um das Produkt zu aktivieren.

Sie können Ihre Lizenz online erwerben und registrieren. Besuchen Sie dazu unseren Web-Shop unter http://www.altova.com/order. Auf dieser Seite erhalten Sie nähere Informationen über Preise (einschl. Mehrfachlizenzrabatte) sowie eine Liste unserer autorisierten Händler und Vertriebspartner.

Wenn Sie XMLSpy 2005 gemeinsam mit anderen Usern verwenden wollen, stellen Sie sicher, dass Sie immer nur das Installationsprogramm weitergeben. Es enthält die Applikation, eine Beschreibung der Grammatik, Beispieldateien, dieses Online-Handbuch und eine kurze ReadMe-Datei in einem einzigen Paket. Jeder, der die XMLSpy 2005-Software von Ihnen erhält, kann die Software 30 Tage lang kostenlos testen. Nach Ablauf des Testzeitraums, muss eine Lizenz erworben werden, damit XMLSpy 2005 weiter verwendet werden kann.

Nähere Informationen finden Sie in der SOFTWARE-PRODUKTLIZENZ am Ende dieses Handbuchs.

2.2 Copyright

All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT are owned by Altova GmbH or its suppliers. The SOFTWARE PRODUCT is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE PRODUCT like any other copyrighted material.

Copyright ©1998-2005 Altova GmbH

All rights reserved.

The Sentry Spelling-Checker Engine Copyright © 2000 Wintertree Software Inc.

STLport Copyright 1999,2000 Boris Fomitchev, Copyright 1994 Hewlett-Packard Company, Copyright 1996,97 Silicon Graphics Computer Systems, Inc., Copyright 1997 Moscow Center for SPARC Technology.

2.3 Software-Produktlizenz

THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS

ALTOVA® END USER LICENSE AGREEMENT

Licensor:

Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Software License Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software ("Software") and any accompanying documentation, including, without limitation printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Software, you agree to be bound by the terms of this Software License Agreement as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below, whether or not you decide to purchase the Software. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Software, and you must destroy any downloaded copies of the Software in your possession or control. Please go to our Web site at http://www.altova.com/eula to download and print a copy of this Software License Agreement for your files and http://www.altova.com/privacy to review the privacy policy.

1. SOFTWARE LICENSE

(a) **License Grant.** Upon your acceptance of this Software License Agreement Altova grants you a non-exclusive, non-transferable (except as provided below), limited license to install and use a copy of the Software on your compatible computer, up to the Permitted Number of computers. The Permitted Number of computers shall be delineated at such time as you elect to purchase the Software. During the evaluation period, hereinafter defined, only a single user may install and use the software on one computer. If you have licensed the Software as part of a suite of Altova software products (collectively, the "Suite") and have not installed each product individually, then the Software License Agreement governs your use of all of the software included in the Suite. If you have licensed SchemaAgent, then the terms and conditions of this Software License Agreement apply to your use of the SchemaAgent server software ("SchemaAgent Server") included therein, as applicable and you are licensed to use SchemaAgent Server solely in connection with your use of Altova Software and solely for the purposes described in the accompanying documentation. In addition, if you have licensed XMLSpy Enterprise Edition or MapForce Enterprise Edition, or UModel, your license to install and use a copy of the Software as provided herein permits you to generate source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile into executable form the complete generated code comprised of the combination of the Restricted Source Code and the Unrestricted Source Code, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or transfer to a third party the Restricted Source Code, unless said third party already has a license to the Restricted Source Code through their separate license agreement with Altova or other agreement with Altova.

Altova reserves all other rights in and to the Software. With respect to the feature(s) of UModel that permit reverse-engineering of your own source code or other source code that you have lawfully obtained, such use by you does not constitute a violation of this Agreement. Except as otherwise permitted in Section 1(h) reverse engineering of the Software is strictly prohibited as further detailed therein.

(b) **Server Use.** You may install one copy of the Software on your computer file server for the purpose of downloading and installing the Software onto other computers within your internal network up to the Permitted Number of computers. If you have licensed .SchemaAgent, then you may install SchemaAgent Server on any server computer or workstation and use it in connection with your Software. No other network use is permitted, including without limitation using the Software either directly or through commands, data or instructions from or to a computer not part of your internal network, for Internet or Web-hosting services or by any user not licensed to use this copy of the Software through a valid license from Altova.

(c) **Concurrent Use**. If you have licensed a "Concurrent-User" version of the Software, you may install the Software on any compatible computers, up to ten (10) times the Permitted Number of users, provided that only the Permitted Number of users actually use the Software at the same time. The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the Software licenses.

(d) **Backup and Archival Copies.** You may make one backup and one archival copy of the Software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the Software. You may not transfer the rights to a backup or archival copy unless you transfer all rights in the Software as provided under Section 3.

(e) **Home Use.** You, as the primary user of the computer on which the Software is installed, may also install the Software on one of your home computers for your use. However, the Software may not be used on your home computer at the same time as the Software is being used on the primary computer.

(f) **Key Codes, Upgrades and Updates.** Prior to your purchase and as part of the registration for the thirty (30) -day evaluation period, as applicable, you will receive an evaluation key code. You will receive a purchase key code when you elect to purchase the Software. The purchase key code will enable you to activate the Software beyond the initial evaluation period. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova. If the Software that you have licensed is an upgrade or an update, then the update replaces all or part of the Software previously licensed. The update or upgrade and the associated license keys does not constitute the granting of a second license to the Software in that you may not use the upgrade or update in addition to the Software that it is replacing. You agree that use of the upgrade of update terminates your license to use the Software or portion thereof replaced.

(g) **Title.** Title to the Software is not transferred to you. Ownership of all copies of the Software and of copies made by you is vested in Altova, subject to the rights of use granted to you in this Software License Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code) and schemas that are authored or created by you via your utilization of the Software, in accordance with its Documentation and the terms of this Software License Agreement, are your property.

(h) **Reverse Engineering.** Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, you may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described

 © 2005 Altova GmbH

herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

(i) **Other Restrictions.** You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except to the limited extent set forth in Section 3 or otherwise expressly provided. You may not copy the Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Software License Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Software License Agreement. You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Software License Agreement and to ensure their compliance with these restrictions. you agree that you are solely responsible for the accuracy and adequacy of the software for your intended use and you will indemnify and HOLD harmless ALTOVA from any 3rd party suit to the extent based upon the accuracy and adequacy of the software in your use. without limitation, The Software is not intended for use in the operation of nuclear facilities, aircraft navigation, communication systems or air traffic control equipment, where the failure of the Software could lead to death, personal injury or severe physical or environmental damage.

2. INTELLECTUAL PROPERTY RIGHTS

Acknowledgement of Altova's Rights. You acknowledge that the Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark. XMLSpy, Authentic, StyleVision, MapForce, Markup Your Mind, Axad, Nanonull, and Altova are trademarks of Altova GmbH (registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows 95, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this Software License Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED TRANSFER RIGHTS

Notwithstanding the foregoing, you may transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer each of this Software License Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; (c) the receiving party secures a personalized key code

from Altova; and (d) the receiving party accepts the terms and conditions of this Software License Agreement and any other terms and conditions upon which you legally purchased a license to the Software. Notwithstanding the foregoing, you may not transfer education, pre-release, or not-for-resale copies of the Software.

4. PRE-RELEASE AND EVALUATION PRODUCT ADDITIONAL TERMS

If the product you have received with this license is pre-commercial release or beta Software ("Pre-release Software"), then this Section applies. In addition, this section applies to all evaluation and/or demonstration copies of Altova software ("Evaluation Software") and continues in effect until you purchase a license. To the extent that any provision in this section is in conflict with any other term or condition in this Software License Agreement, this section shall supersede such other term(s) and condition(s) with respect to the Pre-release and/or Evaluation Software, but only to the extent necessary to resolve the conflict. You acknowledge that the Pre-release Software is a pre-release version, does not represent final product from Altova, and may contain bugs, errors and other problems that could cause system or other failures and data loss. CONSEQUENTLY, THE PRE-RELEASE AND/OR EVALUATION SOFTWARE IS PROVIDED TO YOU **"AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE,** AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE AND/OR EVALUATION SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. If the Evaluation Software has a time-out feature, then the software will cease operation after the conclusion of the designated evaluation period. Upon such expiration date, your license will expire unless otherwise extended. Access to any files created with the Evaluation Software is entirely at your risk. You acknowledge that Altova has not promised or guaranteed to you that Pre-release Software will be announced or made available to anyone in the future, that Altova has no express or implied obligation to you to announce or introduce the Pre-release Software, and that Altova may not introduce a product similar to or compatible with the Pre-release Software. Accordingly, you acknowledge that any research or development that you perform regarding the Pre-release Software or any product associated with the Pre-release Software is done entirely at your own risk. During the term of this Software License Agreement, if requested by Altova, you will provide feedback to Altova regarding testing and use of the Pre-release Software, including error or bug reports. If you have been provided the Pre-release Software pursuant to a separate written agreement, your use of the Software is governed by such agreement. You may not sublicense, lease, loan, rent, distribute or otherwise transfer the Pre-release Software. Upon receipt of a later unreleased version of the Pre-release Software or release by Altova of a publicly released commercial version of the Software, whether as a stand-alone product or as part of a larger product, you agree to return or destroy all earlier Pre-release Software received from Altova and to abide by the terms of the license agreement for any such later versions of the Pre-release Software.

5. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) **Limited Warranty and Customer Remedies.** Altova warrants to the person or entity that first purchases a license for use of the Software pursuant to the terms of this Software License Agreement that (i) the Software will perform substantially in accordance with any accompanying Documentation for a period of ninety (90) days from the date of receipt, and (ii) any support services provided by Altova shall be substantially as described in Section 6 of this agreement. Some states and jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. To the extent allowed by applicable law, implied warranties on the Software, if any, are limited to ninety (90) days. Altova's and its suppliers' entire liability and your exclusive remedy shall be, at Altova's option, either (i) return of the price paid, if any, or (ii) repair or replacement of the Software that does not meet Altova's Limited Warranty and which is returned to Altova with a copy of your receipt. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, misapplication, abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement

 © 2005 Altova GmbH

Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This limited warranty does not apply to Evaluation and/or Pre-release Software.

(b) **No Other Warranties and Disclaimer.** THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR ITS SUPPLIER'S BREACH OF WARRANTY. ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

(c) **Limitation Of Liability.** TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, ALTOVA'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS SOFTWARE LICENSE AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, Altova's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Software License Agreement between Altova and you.

(d) **Infringement Claims.** Altova will indemnify and hold you harmless and will defend or settle any claim, suit or proceeding brought against you by a third party that is based upon a claim that the content contained in the Software infringes a copyright or violates an intellectual or proprietary right protected by United States or European Union law ("Claim"), but only to the extent the Claim arises directly out of the use of the Software and subject to the limitations set forth in Section 5 of this Agreement except as otherwise expressly provided. You must notify Altova in writing of any Claim within ten (10) business days after you first receive notice of the Claim, and you shall provide to Altova at no cost with such assistance and cooperation as Altova may reasonably request from time to time in connection with the defense of the Claim. Altova shall have sole control over any Claim (including, without limitation, the selection of counsel and the right to settle on your behalf on any terms Altova deems desirable in the sole exercise of its discretion). You may, at your sole cost, retain separate counsel and participate in the defense or settlement negotiations. Altova shall pay actual damages, costs, and attorney fees awarded against you (or payable by you pursuant to a settlement agreement) in connection with a Claim to the extent such direct damages and costs are not reimbursed to you by insurance or a third party,

to an aggregate maximum equal to the purchase price of the Software. If the Software or its use becomes the subject of a Claim or its use is enjoined, or if in the opinion of Altova's legal counsel the Software is likely to become the subject of a Claim, Altova shall attempt to resolve the Claim by using commercially reasonable efforts to modify the Software or obtain a license to continue using the Software. If in the opinion of Altova's legal counsel the Claim, the injunction or potential Claim cannot be resolved through reasonable modification or licensing, Altova, at its own election, may terminate this Software License Agreement without penalty, and will refund to you on a pro rata basis any fees paid in advance by you to Altova. THE FOREGOING CONSTITUTES ALTOVA'S SOLE AND EXCLUSIVE LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT. This indemnity does not apply to infringements that would not be such, except for customer-supplied elements.

6. SUPPORT AND MAINTENANCE

Altova offers multiple optional "Support & Maintenance Package(s)" ("SMP") for the version of Software product edition that you have licensed, which you may elect to purchase in addition to your Software license. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. Your rights with respect to support and maintenance as well as your upgrade eligibility depend on your decision to purchase SMP and the level of SMP that you have purchased:

(a) If you have not purchased SMP, you will receive the Software AS IS and will not receive any maintenance releases or updates. However, Altova, at its option and in its sole discretion on a case by case basis, may decide to offer maintenance releases to you as a courtesy, but these maintenance releases will not include any new features in excess of the feature set at the time of your purchase of the Software. In addition, Altova will provide free technical support to you for thirty (30) days after the date of your purchase (the "Support Period" for the purposes of this paragraph a), and Altova, in its sole discretion on a case by case basis, may also provide free courtesy technical support during your thirty (30)-day evaluation period. Technical support is provided via a Web-based support form only, and there is no guaranteed response time.

(b) If you have purchased SMP, then solely for the duration of its delineated Support Period, **you are eligible to receive the version of the Software edition** that you have licensed and all maintenance releases and updates for that edition that are released during your Support Period. For the duration of your SMP's Support Period, you will also be eligible to receive upgrades to the comparable edition of the next version of the Software that succeeds the Software edition that you have licensed for applicable upgrades released during your Support Period. The specific upgrade edition that you are eligible to receive based on your Support Period is further detailed in the SMP that you have purchased. Software that is introduced as separate product is not included in SMP. Maintenance releases, updates and upgrades may or may not include additional features. In addition, Altova will provide Priority Technical Support to you for the duration of the Support Period. Priority Technical Support is provided via a Web-based support form only, and Altova will make commercially reasonable efforts to respond via e-mail to all requests within forty-eight (48) hours during Altova's business hours (MO-FR, 8am UTC – 10pm UTC, Austrian and US holidays excluded) and to make reasonable efforts to provide work-arounds to errors reported in the Software.

During the Support Period you may also report any Software problem or error to Altova. If Altova determines that a reported reproducible material error in the Software exists and significantly impairs the usability and utility of the Software, Altova agrees to use reasonable commercial efforts to correct or provide a usable work-around solution in an upcoming maintenance release or update, which is made available at certain times at Altova's sole discretion.

If Altova, in its discretion, requests written verification of an error or malfunction discovered by you or requests supporting example files that exhibit the Software problem, you shall promptly provide such verification or files, by email, telecopy, or overnight mail, setting forth in reasonable detail the respects in which the Software fails to perform. You shall use reasonable efforts to cooperate in diagnosis or study of errors. Altova may include error corrections in maintenance releases, updates, or new major releases of the Software. Altova is not obligated to

 © 2005 Altova GmbH

fix errors that are immaterial. Immaterial errors are those that do not significantly impact use of the Software. Whether or not you have purchased the Support & Maintenance Package, technical support only covers issues or questions resulting directly out of the operation of the Software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.
Updating Software may require the updating of software not covered by this Software License Agreement before installation. Updates of the operating system and application software not specifically covered by this Software License Agreement are your responsibility and will not be provided by Altova under this Software License Agreement. Altova's obligations under this Section 6 are contingent upon your proper use of the Software and your compliance with the terms and conditions of this Software License Agreement at all times. Altova shall be under no obligation to provide the above technical support if, in Altova's opinion, the Software has failed due to the following conditions: (i) damage caused by the relocation of the software to another location or CPU; (ii) alterations, modifications or attempts to change the Software without Altova's written approval; (iii) causes external to the Software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (iv) your failure to maintain the Software at Altova's specified release level; or (v) use of the Software with other software without Altova's prior written approval. It will be your sole responsibility to: (i) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of Software malfunction and provide Altova with complete information thereof; (ii) provide for the security of your confidential information; (iii) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

7. SOFTWARE ACTIVATION, UPDATES AND LICENSE METERING

(a) **License Metering**. Altova has a built-in license metering module that helps you to avoid any unintentional violation of this Software License Agreement. Altova may use your internal network for license metering between installed versions of the Software.

(b) **Software Activation**. **Altova's Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova license server. You agree that Altova may use these measures and you agree to follow any applicable requirements.**

(c) **LiveUpdate**. Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you.

(d) **Use of Data.** The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this Software License Agreement. By your acceptance of the terms of this Software License Agreement or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Software License Agreement and/or the Privacy Policy as revised from time to time. European users understand and consent to the processing of personal information in the United States for the purposes described herein. Altova has the right in its sole discretion to amend this provision of the Software License Agreement and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

8. TERM AND TERMINATION

This Software License Agreement may be terminated (a) by your giving Altova written notice of termination; or (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Software License Agreement and fail to cure such breach within ten (10) days after notice from Altova. In addition the Software License Agreement governing your use of a previous version that you have upgraded or updated of the Software is terminated upon your

acceptance of the terms and conditions of the Software License Agreement accompanying such upgrade or update. Upon any termination of the Software License Agreement, you must cease all use of the Software that it governs, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software remain in your possession or control. The terms and conditions set forth in Sections 1(g), (h), (i), 2, 5(b), (c), 9, and 10 survive termination as applicable.

9. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS.

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. §12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. §227-7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this Software License Agreement. Manufacturer is Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

10. GENERAL PROVISIONS

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Software License Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software in the United States then this Software License Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the federal or state courts of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Software License Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This Software License Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

This Software License Agreement contains the entire agreement and understanding of the parties

 © 2005 Altova GmbH

with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this Software License Agreement shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova's Web site for Altova and the address shown in Altova's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This Software License Agreement will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party's right at a later time to enforce the same or any other term of this Software License Agreement. This Software License Agreement may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this Software License Agreement by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this Software License Agreement. If, for any reason, any provision of this Software License Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Software License Agreement, and this Software License Agreement shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2005-05-05

Index

▪

.NET, 206
Integration von XMLSPY mit, 206
und SOAP Debugger, 214
und XSLT Debugger, 211
Unterschiede zur XMLSPY Standalone-Version, 207

A

Abfrage, 466
Oracle XML DB, 466
für DB-Anzeige in der Authentic-Ansicht, 196
Absatzstil, 402
Access, 391, 448, 449
Datenbank - Schema erzeugen, 404
ActiveX Data Object, 391
ADO, 390, 391, 448, 449
Connection String, 395
Data Link Eigenschaften, 395
ADO connection string, 81
ADOX, 448, 449
Adresse, 98
mit einem Binding verknüpfen (WSDL), 98
Aktiviere Versionskontrolle, 314
Aktivieren, 535
Lizenz, 535
Aktuelle Execution Nodes einblenden, 382
XSLT Debugger, 382
Aktuellste Version holen, 315
Alle schließen, 292
Alles markieren, 302
Befehl, 302
Altova, 6
Produktfamilie, 6
Altova XQuery Prozessor, 542
Altova XSLT 1.0 Prozessor, 538
Altova XSLT 2.0 Prozessor, 539
Anbieter, 310
Versionskontrolle, 310
Änderungen verfolgen, 518
Anforderungen, 841
Angedocktes Fenster, 532
Anhang, 846
Anhängen, 334
Operation an PortType, 98
Zeile (Dokument Editor), 388
Anpassen, 511
Kontextmenü, 511
Kontextmenü 'Anpassen', 516
Makros, 513
Menü, 511
Symbolleiste/Menü Befehle, 504
Ansicht, 474, 475, 476
Ansicht, 475
ausklappen, 474
Browser-Ansicht, 474
einklappen, 475
Enhanced Grid-Ansicht, 473
erweitern, 475
Gehe zu Datei, 476
Gehe zu Zeile/Zeichen, 476
Menü, 473
Optimale Breite, 475
reduzieren, 475
Schema Design-Ansicht, 474
Textansicht, 473
XMLSpy in Eclipse, 230
Dialogfeld in der Content Model-Ansicht, 364
Ansicht wechseln, 177
zur Authentic-Ansicht, 177
Anwendung beenden, 297
Anzeige-Einstellungen, 526
Anzeigen, 532, 533
API, 596
documentation, 596
JAVA, 801
JAVA Classpath, 801
overview, 597
Application, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631
ActiveDocument, 620
AddMacroMenuItem, 621
Application, 621
ClearMacroMenu, 621
CurrentProject, 621
Dialogs, 622
Documents, 622

© 2005 Altova GmbH

Application, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631
GetDatabaseImportElementList, 622
GetDatabaseSettings, 623
GetDatabaseTables, 623
GetExportSettings, 624
GetTextImportElementList, 624
GetTextImportExportSettings, 625
ImportFromDatabase, 625
ImportFromSchema, 626
ImportFromText, 627
ImportFromWord, 628
NewProject, 628
OnBeforeOpenDocument, 618
OnBeforeOpenProject, 619
OnDocumentOpened, 619
OnProjectOpened, 620
OpenProject, 628
Parent, 629
Quit, 629
ShowApplication, 629
ShowForm, 630
URLDelete, 630
URLMakeDirectory, 630
WarningNumber, 631
WarningText, 631
Arabisch, 844
Arbeitsspeicher, 841
Arbeitsverzeichnis, 312
Versionskontrolle, 312
Arial Unicode MS, 843
Assigning, 549
scripting projects, 549
ATL, 586
plug-in sample files, 586
ATTLIST Deklaration, 341
als Child in Grid-Ansciht hinzufügen, 341
in Grid-Ansicht anhängen, 338
in Grid-Ansicht einfügen, 334
Konvertieren in (Grid-Ansicht), 343
Attribut, 839
Import, 77
in Grid-Ansicht anhängen, 335
Konvertieren in (Grid-Ansicht), 342
Attribute, 331
als Child in Grid-Ansicht hinzufügen, 339
in der Content Model-Ansicht ein- und ausblenden, 43
in der Schema-Übersicht für eine Komponente definieren, 142
in Grid-Ansicht einfügen, 331
in Schemadefinitionen, 43
Attribute standardmäßig, 391
AttributeFormDefault, 361
Einstellungen in der Schema Design-Ansicht, 361
AttributeGroup, 839
Attribut-Eingabehilfe, 183
in der Authentic-Ansicht, 183
Verwendung in der Authentic-Ansicht, 94
Attribut-Vorschau, 523
Attributwerte, 94
in der Authentic-Ansicht eingeben, 94
Ausblenden, 388
Markup, 388
Auschecken, 315
Auschecken rückgängig, 316
Ausgabeformatierung, 518
Ausgliedern, 319
aus Versionskontrolle, 319
Ausschließen, 391
Primär-/Sekundärschlüssel, 391
Ausschneiden, 298
Befehl, 298
Aussteigen, 381
XSLT Debugger, 381
Authentic, 385
DB-Daten bearbeiten, 385
interface, 173
Menü, 384
Projektfenster, 175
Anzeige der Markup-Symbole, 177, 180
Anzeige des Dokuments, 180
Attribut-Eingabehilfe, 183
Attributwerte eingeben, 94
Beispiel für statische Tabelle, 90
Daten eingeben, 89
Dateneingabe-Steuerelemente, 90
dynamische Tabellen bearbeiten, 177
Eingabehilfen, 175, 183
Element anwenden (ersetzen), 94
Element einfügen, 93
Element löschen, 94
Elemente hinzufügen, 90
Element-Eingabehilfe, 183
Elementinhalt hinzufügen, 90
Entities einfügen, 95

© 2005 Altova GmbH

Authentic, 385
Entities-Eingabehilfe, 183
Hauptfenster, 175, 180
Kontextmenü, 183
Kontextmenüs, 183, 187
Menüleiste, 175
neue XML-Datei öffnen, 384
SPS-Tabellen, 188
Statusleiste, 175
Symbolleiste, 175
Symbolleistenschaltflächen, 177
Tabellen (SPS und XML), 188
Text eingeben, 90
Text formatieren, 177
Tutorial, 88
Übersicht über die GUI, 175
Verwendung von XML-Tabellen, 189
Verwendung wichtiger Features, 188
Voraussetzungen für das Tutorial, 88
Vorlage, 88
wechseln zur, 177, 474
XML-Dokument drucken, 96
XML-Dokument öffnen, 88
XML-Tabellen, 189
XML-Tabellensymbole, 193
XPath zum ausgewählten Node, 175

AuthenticDataTransfer, 632, 633
dropEffect, 632
getData, 633
ownDrag, 633
type, 633

Authentic-Menü, 177
Anzeige von Markup-Symbolen, 177
dynamische Tabellen bearbeiten, 177

AuthenticRange, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660
AppendRow, 636
Application, 636
CanPerformAction, 636
CanPerformActionWith, 637
Close, 637
CollapsToBegin, 638
CollapsToEnd, 638
Copy, 638
Cut, 638
Delete, 639
DeleteRow, 639
DuplicateRow, 640
ExpandTo, 640
FirstTextPosition, 640
FirstXMLData, 641
FirstXMLDataOffset, 642
GetElementAttributeNames, 643
GetElementAttributeValue, 643
GetElementHierarchy, 644
GetEntityNames, 644
Goto, 645
GotoNext, 645
GotoNextCursorPosition, 646
GotoPrevious, 646
GotoPreviousCursorPosition, 647
HasElementAttribute, 647
InsertEntity, 647
InsertRow, 648
IsCopyEnabled, 648
IsCutEnabled, 649
IsDeleteEnabled, 649
IsEmpty, 649
IsEqual, 649
IsFirstRow, 650
IsInDynamicTable, 650
IsLastRow, 650
IsPasteEnabled, 651
IsTextStateApplied, 651
LastTextPosition, 651
LastXMLData, 652
LastXMLDataOffset, 653
MoveBegin, 654
MoveEnd, 654
MoveRowDown, 655
MoveRowUp, 655
Parent, 655
Paste, 656
PerformAction, 656
Select, 657
SelectNext, 657
SelectPrevious, 658
SetElementAttributeValue, 659
SetFromRange, 660
Text, 660

AuthenticView, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675
Application, 669
AsXMLString, 669
DocumentBegin, 670

© 2005 Altova GmbH

AuthenticView, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675
DocumentEnd, 670
Event, 670
Goto, 670
IsRedoEnabled, 671
IsUndoEnabled, 671
MarkupVisibility, 672
OnBeforeCopy, 662
OnBeforeCut, 662
OnBeforeDelete, 663
OnBeforeDrop, 663
OnBeforePaste, 664
OnDragOver, 665
OnKeyboardEvent, 666
OnMouseEvent, 667
OnSelectionChanged, 668
Parent, 672
Print, 672
Redo, 673
Selection, 673
Undo, 673
WholeDocument, 674
XMLDataRoot, 675
Autokomplettierung, 116
in der Textansicht, 116
Auto-Komplettierung, 522
Text-Ansicht Ein/Aus, 522
Automatisch, 428
Erzeugen des Shape string, 428
Automatische Validierung, 520
Automatische Validierung ausschalten, 520

B

Bearbeiten, 516
Makro Schaltfläche, 516
Menü, 298
Bearbeiten rückgängig, 316
Bearbeiten von Inhalt, 89, 90
in der Authentic-Ansicht, 89, 90
Bearbeitet mit XMLSPY, 518
Beenden, 381
Debuggersitzung, 381
Befehl, 504
hinzufügen zu Symbolleiste/Menü, 504
Kontextmenü, 511
löschen aus Menü, 511
Menü zurücksetzen, 511
Auflistung in Tastaturzuordnung, 534
Beispieldateien, 17
Tutorial, 17
Benutzerdefiniert, 490
Wörterbuch, 490
Benutzerhandbuch, 1
Benutzerhandbücher, 6
für Altova-Produkte, 6
Benutzeroberfläche, 11
Übersicht, 11
Bestellformular, 536
Bestellung, 847
Big-endian, 528
Binding, 100
definieren (WSDL), 100
BizTalk, 838, 839
Breakpoint, 382
Dialogfeld, 382
Breakpoint Aus - XSLT Debugger, 382
Breakpoint Ein - XSLT Debugger, 382
Breakpoints, 248
im XSLT/XQuery Debugger, 248
Breakpoints dialog, 235
Broadcast, 845
Browser, 523
Ansicht, 474
Menü, 478
Beschreibung, 205
Inhalt aktualisieren, 478
Laden einer Seite abbrechen, 478
neues Fenster, 478
Schriftgröße, 478
weiter, 478
zurück, 478

C

C#, 262
Code, 262
C++, 841
Code, 262
Call Stack-Fenster, 244
in XSLT/XQuery Debugger, 244

© 2005 Altova GmbH

CDATA, 340
als Child in Grid-Ansicht hinzufügen, 340
in Grid-Ansicht anhängen, 336
in Grid-Ansicht einfügen, 332
Konvertieren in (Grid-Ansicht), 342
Child anhängen, 338
in Grid-Ansicht, 338
Class, 801
JAVA, 801
ClassPath statement, 801
Client, 412
Oracle, 410
Code, 280
integrierte Typen, 280
SPL, 274
Code Generator, 262, 360
CodeGeneratorDlg, 676, 677, 678
Application, 676
CPPSettings_DOMType, 676
CPPSettings_LibraryType, 676
CPPSettings_UseMFC, 676
CSharpSettings_ProjectType, 677
OutputPath, 677
OutputPathDialogAction, 677
OutputResultDialogAction, 677
Parent, 678
ProgrammingLanguage, 678
PropertySheetDialogAction, 678
TemplateFileName, 678
Codepage, 524
COM, 845
documentation, 596
Comment, 340
als Child in Grid-Ansicht hinzufügen, 340
in Grid-Ansicht anhängen, 336
in Grid-Ansicht einfügen, 332
Konvertieren in (Grid-Ansicht), 343
ComplexType, 839
Definition erweitern, 34
in Schema-Definitionen, 34
Configure, 583
XMLSPY UI, 583
Connection String, 391, 449
ADO - Datenbankschema erzeugen, 81
Constraints, 81
Identity, 81
Content Model, 839
Attribute ein- und ausblenden, 43
Diagramm speichern, 362
einfaches Content Model erstellen, 23
Content Model-Ansicht, 19
ausführliche Beschreibung, 146
Grundzüge, 137
konfigurieren, 364
Content Models, 146
von Schema-Komponenten, 146
Copyright, 846, 848, 849
CPU, 841
CR&LF, 518
create, 630
CSS, 402
CVS, 310

D

Darstellung, 523
Data Access Komponenten, 391
Data Link Eigenschaften, 395
Database, 81
OLE DB Anbieter, 81
SQL ,Oracle etc., 81
DatabaseConnection, 679, 680, 681, 682
ADOConnection, 679
AsAttributes, 680
CreateMissingTables, 680
CreateNew, 680
DatabaseKind, 680
ExcludeKeys, 681
File, 681
IncludeEmptyElements, 681
NumberDateTimeFormat, 681
ODBCConnection, 682
SQLSelect, 682
TextFieldLen, 682
Datei, 286
Alle schließen, 292
Alles speichern, 294
als Mail senden, 294
Grid-Ansicht drucken ,Druckoptionen, 295
in Eclipse importieren, 221
Kodierung, 291
Menü, 286
Neu laden, 291
Öffnen, 289

© 2005 Altova GmbH

Datei, 286
Optionen beim Öffnen, 518
Register, 518
schließen, 292
speichern, 292
Speichern unter, 292
Speichern unter URL, 292
Standard-Encoding, 528
URL öffnen, 289
Versionskontrolle, 315
Datei wählen, 390
Dateiarten, 520
Dateien, 501
Vergleichsoptionen, 501
zuletzt verwendete, 297
zwei vergleichen, 497
Kodierung, 291
Dateien vergleichen, 497
Optionen, 501
Dateierweiterungen, 119
für XQuery-Dateien, 119
Daten, 132
in und aus externen Applikationen importieren und exportieren, 132
Datenaustausch, 390, 391, 402, 448, 449, 450
Datenbank, 390, 391, 449
Attribut importieren, 77
DB erzeugen basierend auf Schema, 432
Element importieren, 77
Export, 448
Export von XML-Daten, 74
Exportrestriktionen, 74
Feld überspringen, 77
Hierarchie - keyref, 81
Oracle XML DB, 466
Primär- / Sekundärschlüssel, 77
Schema erzeugen, 403
Schema erzeugen von, 81
in der Enhanced Grid-Ansicht, 132
Datenbank/Tabellenansicht, 64
Verwendung, 64
Datenbanken, 385
in Authentic-Ansicht bearbeiten, 385
siehe auch DB, 195
Datenbankimport, 395
ADO, 395
Dateneingabe-Steuerelemente, 90
in der Authentic-Ansicht, 90
Datenformat, 448
Datensatz, 391
Datumswahl, 201
in der Authentic-Ansicht verwenden, 201
DB, 196
Abfragen erstellen, 196
Abfragen in der Authentic-Ansicht, 195
Anzeige in der Authentic-Ansicht filtern, 196
Bearbeiten in der Authentic-Ansicht, 195
in der Authentic-Ansicht bearbeiten, 200
Navigieren in Tabellen in der Authentic-Ansicht, 195
Parameters in DB-Abfragen, 196
DB Provider, 391
DB2, 412
Datenbank - Schema erzeugen, 412
DB-Struktur, 417
Schema Extensions, 417
Schemaoptionen, 415
DBUri, 466
Datenbank abfragen, 466
DCD, 838, 839
Deaktiviere Versionskontrolle, 314
Debugger, 382
aktuelle Execution Nodes einblenden, 382
aussteigen, 381
Breakpoint Ein/Aus, 382
Breakpoint einfügen/löschen, 382
Debug-Fenster, 383
Dialogfeld Breakpoints, 382
einsteigen, 381
Einstellungen, 383
Sitzung beenden, 381
SOAP Debugger, 487
SOAP in Eclipse, 227
SOAP Kommunikation - Beispiel, 253
überspringen, 382
XSLT Debugger erneut starten, 381
XSLT Debugger starten, 380
XSLT Debugger stoppen, 381
XSLTin Eclipse, 225
Anordnung, 246
delete, 630
Details-Eingabehilfe, 23
Diagramm anzeigen, 368
Dialogs, 683, 684
Application, 683
CodeGeneratorDlg, 683
FileSelectionDlg, 683

© 2005 Altova GmbH

Dialogs, 683, 684
Parent, 684
SchemaDocumentationDlg, 684
directory, 630
Diskussionsforen, 6
für XML-Probleme, 6
DocEditEvent (obsolete), 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759
altKey (obsolete), 745
altLeft (obsolete), 746
button (obsolete), 747
cancelBubble (obsolete), 748
clientX (obsolete), 749
clientY (obsolete), 750
ctrlKey (obsolete), 751
ctrlLeft (obsolete), 752
dataTransfer (obsolete), 753
fromElement (obsolete), 754
keyCode (obsolete), 754
propertyName (obsolete), 755
repeat (obsolete), 755
returnValue (obsolete), 755
shiftKey (obsolete), 756
shiftLeft (obsolete), 757
srcElement (obsolete), 758
type (obsolete), 759
DocEditView (obsolete), 765, 766, 767, 768, 769, 770, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789
ApplyTextState (obsolete), 765
CurrentSelection (obsolete), 766
EditClear (obsolete), 766
EditCopy (obsolete), 767
EditCut (obsolete), 767
EditPaste (obsolete), 768
EditRedo (obsolete), 768
EditSelectAll (obsolete), 769
EditUndo (obsolete), 769
event (obsolete), 770
GetAllowedElements (obsolete), 770
GetNextVisible (obsolete), 773
GetPreviousVisible (obsolete), 774
IsEditClearEnabled (obsolete), 775
IsEditCopyEnabled (obsolete), 775
IsEditCutEnabled (obsolete), 776
IsEditPasteEnabled (obsolete), 776
IsEditRedoEnabled (obsolete), 777
IsEditUndoEnabled (obsolete), 777
IsRowAppendEnabled (obsolete), 778
IsRowDeleteEnabled (obsolete), 778
IsRowDuplicateEnabled (obsolete), 779
IsRowInsertEnabled (obsolete), 779
IsRowMoveDOwnEnabled (obsolete), 780
IsRowMoveUpEnabled (obsolete), 780
IsTextStateApplied (obsolete), 781
IsTextStateEnabled (obsolete), 781
LoadXML (obsolete), 782
RowAppend (obsolete), 783
RowDelete (obsolete), 783
RowDuplicate (obsolete), 784
RowInsert (obsolete), 784
RowMoveDown (obsolete), 785
RowMoveUp (obsolete), 785
SaveXML (obsolete), 786
SelectionMoveTabOrder (obsolete), 787
SelectionSet (obsolete), 788
XMLRoot (obsolete), 789
DOCTYPE Deklaration, 340
als Child in Grid-Ansicht hinzufügen, 340
in Grid-Anischt anhängen, 336
in Grid-Ansicht einfügen, 332
Konvertieren in (Grid-Ansicht), 343
Document, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705
Application, 688
AssignDTD, 688
AssignSchema, 688
AssignXSL, 689
AssignXSLFO, 689
AuthenticView, 689
Close, 690
ConvertDTDOrSchema, 690
CreateChild, 691
CreateSchemaDiagram, 691
CurrentViewMode, 691
DataRoot, 692
DocEditView, 692
Encoding, 692
EndChanges, 693
ExecuteXQuery, 695
ExportToDatabase, 693
ExportToText, 694
FullName, 695
GenerateDTDOrSchema, 695
GenerateProgramCode, 696
GenerateSchemaDocumentation, 696

© 2005 Altova GmbH

Document, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705
GetExportElementList, 697
GetPathName, 697
GridView, 697
IsModified, 698
IsValid, 698
IsWellFormed, 699
Name, 700
OnBeforeCloseDocument, 686
OnBeforeSaveDocument, 686
OnCloseDocument, 687
OnViewActivation, 687
Path, 700
RootElement, 700
Save, 701
SaveAs, 701
Saved, 701
SaveInString, 702
SaveToURL, 702
SetActiveDocument, 702
SetEncoding, 703
SetPathName, 704
StartChanges, 704
SwitchViewMode, 704
Title, 705
TransformXSL, 705
TransformXSLFO, 705
UpdateViews, 705
XQuery, 695
Document Content Description, 839
Document Type Definition, 839
Documents, 706, 707, 708
Count, 706
Item, 707
NewFile, 707
NewFileFromText, 707
OpenFile, 707
OpenURL, 708
OpenURLDialog, 708
Dokument, 471
Hinzufügen von Instanzen in Oracle XML DB, 471
Oracle XML DB durchsuchen, 470
Rechschreibprüfung, 490
für Altova-Produkte, 6
von XML-Schema, 362
WSDL, 482
WSDL erstellen, 110
Dokumente bearbeiten, 130
in der Grid-Ansicht, 130
Dokumentieren, 47
Schema, 47
Download, 536
Komponenten, 536
Download Versionskontroll-Projekt, 312
Drag & Drop, 27
Elemente hinzufügen, 27
Draw menu, 579
in Scripting Environment, 579
Druckansicht, 296
Druckaufbereitung, 301
Text-Ansicht, 301
Drucken, 96
von der Authentic-Ansicht, 96
Druckereinrichtung, 296
DTD, 518, 520, 838, 839
Attlist-Deklaration, 334, 338, 341, 343
Code generieren von, 360
Element-Deklaration, 334, 338, 341, 343
Entities inkludieren, 354
Entity-Deklaration, 334, 338, 341, 343
Entwurf anhand von XML-Datei erzeugen, 359
Gehe von XML-Dokument zu Definition, 355
Gehe von XML-Dokument zu DTD, 355
Konvertieren in XML-Schema, 356
Menübefehle zu, 354
Notation Deklaration, 343
Notation-Deklaration, 334, 338, 341
Referenz von XML zu externer DTD, 333
von XML-Dokument generieren, 355
zu XML-Dokument zuweisen, 354
Menü, 354
Durchsuchen, 470
Oracle XML DB, 470
Oracle XML DB abfragen, 466
Dynamische (SPS) Tabellen in der Authentic-Ansicht, 188
Verwendung, 188
Dynamische Tabellen, 177
bearbeiten, 177

© 2005 Altova GmbH

E

Eclipse, 221
Dateien importieren, 221
starten und Projekt definieren, 217
und SOAP Debugger, 227
und XMLSpy, 216
und XSLT Debugger, 225
Unterschiede zu XMLSpy, 224
XML-Datei erstellen, 219
Edit menu, 574
in Scripting Environment, 574
Eigenschaften, 470
Oracle XML DB, 470
Versionskontrolle, 321
Einchecken, 316
Einfügen, 299
Befehl, 299
Breakpoint - XSLT Debugger, 382
in Grid-Ansicht, 331
Zeile (Dokument Editor), 388
in der Authentic-Ansicht, 93
Einführung, 263
Code Generator, 263
Eingabehilfe, 532, 533
Details, 23
Eingabehilfen, 17
aktualisieren, 352
für XQuery, 120
Grid-Ansicht, 126
in der Textansicht, 114
in Grid-Ansicht, 63
in Schema-Ansicht, 137
in WSDL Design-Ansicht, 169
Befehl, 352
Eingabemethoden, 843
Eingabetaste, 90
Auswirkung bei Drücken, 90
drücken, 204
Einklappen, 477
Child-Nodes ausklappen (Textansicht), 477
Einrücken, 301
in Text-Ansicht, 301
Einrücklinien, 113
in Textansicht, 113
Einschränkungen, 74
Export, 74
Einsteigen, 381
XSLT Debugger, 381
Einstellungen, 518
Skripting, 530
XSLT Debugger, 383
Einstellungen für den Dateivergleich, 501
Einzellizenz, 535
Einzelplatzlizenz, 845
Electronic Software Distribution, 847
Element, 839
als Child in Grid-Ansicht hinzufügen, 339
Import, 77
in Grid-Ansicht anhängen, 335
in Grid-Ansicht einfügen, 331
Konvertieren in (Grid-Ansicht), 342
in der Authentic-Ansicht, 94
ELEMENT Deklaration, 341
als Child in Grid-Ansicht hinzufügen, 341
Konvertieren in (Grid-Ansicht), 343
Element ersetzen, 94
in der Authentic-Ansicht, 94
Element löschen, 94
in der Authentic-Ansicht, 94
ELEMENT-Deklaration, 338
in Grid-Ansicht anhängen, 338
in Grid-Ansicht einfügen, 334
Elemente, 30
Inhalt einschränken, 30
optional machen, 30
in Textansicht, 113
Elemente reduzieren, 113
in Textansicht, 113
Elemente standardmäßig, 391
Element-Eingabehilfe, 183
in der Authentic-Ansicht, 183
Verwendung in der Authentic-Ansicht, 90, 93, 94
ElementFormDefault, 361
Einstellungen in der Schema Design-Ansicht, 361
ElementList, 709, 710
Count, 709
Item, 710
RemoveElement, 710
ElementListItem, 710, 711
ElementKind, 710
FieldCount, 710
Name, 711

© 2005 Altova GmbH

ElementListItem, 710, 711
RecordCount, 711
Elementtyp, 53
in XML-Dokument festlegen, 53
E-mail, 535
Dateien senden, 294
Encoding, 291
Standard, 528
Enhanced Grid-Ansicht, 473
Beschreibung, 126
siehe Grid-Ansicht, 55
Entities, 202
in der Authentic-Ansicht definieren, 202
in der Authentic-Ansicht Entities einfügen, 95
in der Authentic.Ansicht, 183
Verwendung in der Authentic-Ansicht, 95
ENTITY Deklaration, 341
als Child in Grid-Ansicht hinzufügen, 341
in Grid-Ansicht anhängen, 338
Konvertieren in (Grid-Ansicht), 343
Entity-Deklaration, 334
in Grid-Ansicht einfügen, 334
Entkoppeln, 497
Skripts von Projekt, 497
Entsperren, 535
Enumerationen, 43
für Attribute definieren, 43
Enumerations, 791, 792, 793, 794, 795, 796, 797, 798, 799
SPYAuthenticActions, 791
SPYAuthenticDocumentPosition, 791
SpyAuthenticElementActions, 791
SPYAuthenticElementKind, 792
SPYAuthenticMarkupVisibility, 792
SPYDatabaseKind, 792
SPYDialogAction, 793
SPYDOMType, 793
SPYDTDSchemaFormat, 793
SPYEncodingByteOrder, 793
SPYExportNamespace, 793
SPYFrequentElements, 794
SPYKeyEvent, 794
SPYLibType, 794
SPYLoading, 794
SPYMouseEvent, 794
SPYNumberDateTimeFormat, 795
SPYProgrammingLanguage, 795
SPYProjectItemTypes, 796
SPYProjectType, 796
SPYSchemaDefKind, 796
SPYSchemaDocumentationFormat, 797
SPYTextDelimiters, 797
SPYTextEnclosing, 798
SPYTypeDetection, 798
SPYURLTapes, 798
SPYViewModes, 798
SPYVirtualKeyMask, 799
SPYXMLDataKind, 799
Erneut starten, 381
XSLT Debugger, 381
Ersetzen, 305
Text in Dokument, 305
Erstellen, 432
DB basierend auf Schema, 432
leere Elemente bei Import, 391
Neues XML Dokument, 51
Service und Port, 98
WSDL-Dokumentation, 110
XML-Datei in Eclipse, 219
Erweitern, 475
alles, 475
Child-Nodes einklappen (Textansicht), 477
Erweiterte Validierung, 166
in SchemaAgent, 373
Erzeugen, 403
Datenbank-Schema, 403
Datenbankschema (Access), 404
Datenbankschema (Oracle), 410
Datenbankschema (SQL), 407
Datenbankstruktur basierend auf Schema, 432
Schema (IBM DB2), 412
ESD, 847
Evaluierungsversion, 535
Event, 618, 619, 620, 662, 663, 664, 665, 666, 667, 668, 686, 687, 715, 716, 717
in Scripting Environment, 562
invoking, 551
Events, 601
Explorer, 520
Export, 390, 448
Daten aus Datenbanken, 74
Datenbank, 449
Einschränkungen, 74
Parameter, 74
Textdatei, 450
ExportSettings, 711, 712, 713

© 2005 Altova GmbH

ExportSettings, 711, 712, 713
CreateKeys, 711
ElementList, 712
EntitiesToText, 712
ExportAllElements, 712
FromAttributes, 712
FromSingleSubElements, 712
FromTextValues, 713
IndependentPrimaryKey, 713
Namespace, 713
SubLevelLimit, 713
Extensions, 417
Schema erzeugen, 417
Extern geparste Entitäten, 520
ExternalID Deklaration, 340
als Child in Grid-Ansicht hinzufügen, 340
in Grid-Ansicht anhängen, 337
in Grid-Ansicht einfügen, 333
Konvertieren in (Grid-Ansicht), 343
Externer XSL Prozessor, 529
Extras, 490
Menü, 490

F

Facets, 839
FAQ, 391, 536
auf der Altova Website, 6
Farbe, 524, 525
Register, 526
Tabelle, 526
Favoriten, 534
Fehlerbehebung, 6
Online-Hilfe und Support, 6
Fenster, 533
Eingabehilfe, 532, 533
horizontal nebeneinander, 532
Info, 532, 533
Menü, 532
öffnen, 533
Projekt, 532, 533
überlappend, 532
Übersicht, 17
vertikal anordnen, 532
File menu, 574
in Scripting Environment, 574
FileSelectionDlg, 714, 715
Application, 714
DialogAction, 714
FullName, 714
Parent, 715
Finden, 305
siehe Suchen, 305
Flaches Schema, 404
SQL/XML Standard, 404
Foren, 6
zur Diskussion von XML-Problemen, 6
Form Editor, 557
description of, 557
Form Objects, 556
in Scripting Environment, 556
Format für Datenausgabe, 448
Forms, 564
creating in Scripting Environment, 564
designing with Form Editor, 557
in scripting projects, 549
running in application, 551
Fragen und Antworten, 536
Funktionstasten, 507

G

Garantie, 849
Gehe zu Datei, 476
Gehe zu Zeile/Zeichen, 476
Gemischte Markups, 388
anzeigen, 388
Generieren, 262
Code von Schema, 262
WSDL-Dokumentation, 110
Gleichzeitige Verwendung, 535
Global, 843
Einstellungen, 518
in XML-Schemas verwenden, 41
Globale Elemente anzeigen, 368
Globale Komponenten, 137
Auflistung in GUI, 137
in der Schema-Übersicht erstellen, 142
Grafikformate, 204
in der Authentic-Ansicht, 204
Grammatik, 520
Graue Leiste, 474, 475

© 2005 Altova GmbH

Grid View, 132
Datenbank/Tabellen-Ansicht, 132
Grid-Ansicht, 473, 475
Attribut / Element anhängen, 63
bearbeiten, 346
Beschreibung, 126
Dateneingaben, 55
editieren in, 130
Eingabehilfen, 126
Eingabehilfen verwenden, 63
und Tabellenansicht, 64
zu Tabellenansicht wechseln, 344
Grid-Schriftarten, 524
GridView, 715, 716, 717, 718
CurrentFocus, 717
Deselect, 718
IsVisible, 718
OnBeforeDrag, 715
OnBeforeDrop, 716
OnBeforeStartEditing, 716
OnEditingFinished, 717
OnFocusChanged, 717
Select, 718
SetFocus, 718
Größe, 525
Große Markups, 388
anzeigen, 388
Große Markup-Symbole anzeigen, 177, 180
GUI, 11
Übersicht, 11

H

Haftung, 849
Haftungsbeschränkungen, 849
Händler, 847
Hauptfenster, 17
Hebräisch, 844
Hilfe, 534
Index, 534
Inhaltsverzeichnis, 534
Menü, 534
Suchen, 534
Tastaturzuordnungen, 534
Hilfesystem, 534
Hin- und Herschalten, 532, 533
Hintergrundinformationen, 837
Hin-und Herschalten, 532
Hinzufügen, 322
Dateien zu Projekt, 322
Ordner zu Projekt, 323
Projekt in Versionskontrolle, 317
Versionskontrolle, 317
Hiragana, 843
Höher stufen, 346
Befehl, 346

I

IANA, 845
IBM, 412
DB2-Schema erzeugen, 412
ICComNet, 845
Identity Constraint, 43
in der Content Model-Ansicht ein- und ausblenden, 43
Identity Constraints, 81
in der Schema-Übersicht für eine Schema-Komponente definieren, 142
Import, 390
Abschnitt aus Textspalte, 390
alle als Attribute, 391
alle als Elemente, 391
Attribut, 77
Auto Root Element, 77
Daten aus Datenbanken, 77
Datenbank, 395
Datenbankdaten, 391
Element, 77
Erstellen leerer Elemente, 391
Feld überspringen, 77
partiell hierarchische Datensätze, 428
Primär- / Sekundärschlüssel, 77
Statement wählen, 77
Textdatei, 390
Word Dokument, 402
XML basierend auf Schema, 428
In Element einbinden, 346
Befehl, 346
Index, 534
Hilfe, 534
Info, 17
Fenster, 17

© 2005 Altova GmbH

Infofenster, 17
Info-Fenster, 532, 533
Info-Fenster in XSLT/XQuery Debugger, 246
Informationsfenster, 246
anordnen, 246
Inhaltsverzeichnis, 534
Input, 100
Message zu Operation hinzufügen, 100
Installation, 535
Instanzen, 23
Anzahl, 23
Intelligentes Editieren, 522
in der Textansicht, 116
Internet, 535, 536, 537, 842
Internet Explorer, 840
ISO/IEC 10646, 843

J

JAVA, 801
API, 801
ClassPath, 801
Code, 262
Jet Engine, 391, 448, 449

K

Kanji, 843
Kapitel, 534
Katakana, 843
Key, 403
identity constraints, 403
Key-code, 535, 847
Keyref, 403
identity constraints, 403
key - Datenbank Hierarchie, 81
Klappleiste, 113
in Textansicht, 113
Kleine Markups, 388
anzeigen, 388
Kleine Markup-Symbole anzeigen, 177, 180
Kodierung, 291
von Dateien, 291
Kommunikationsvorgang, 252
SOAP Debugger, 252
Komponenten, 137
Auflistung in GUI, 137
Komponenten downloaden, 536
Komponenten-Definitionen, 34
wiederverwenden, 34
Komponenten-Navigator, 137
Beschreibung, 137
Kompositor, 23
für Sequences, 23
in der Schema-Ansicht, 146
Kontextfenster, 243
im XSLT/XQuery Debugger, 243
Kontextmenü, 516
anpassen, 516
Befehle, 511
in der Authentic-Ansicht, 183, 187
Konvertieren, 449
Datenbank exportieren, 449
Erstellen einer Datenbank von Schema, 81
Export in Textdateien, 448
Exportieren in Text / Datenbank, 74
Import von Daten aus Datenbanken, 77
Importieren von Datenbankdaten, 395
Menü, 390
Oracle XML DB, 466
Schema in Datenbankstruktur, 432
Textdatei exportieren, 450
in Grid-Ansicht, 342
Kopieren, 299
als strukturierten Text, 299
als XML-Text, 299
Befehl, 298
XPath, 301
Zeile (Dokument Editor), 388
Kopieren als strukturierten Text, 299
Kurse, 537
Training, 537
Kürzel, 507
anzeigen in Tooltip, 516
zuweisen/löschen, 507

L

LAN, 845

© 2005 Altova GmbH

Layers, 557
in design of Forms with Form Editor, 557
Layout menu, 576
in Scripting Environment, 576
Layout of Forms, 557
in Form Editor, 557
Leere Elemente, 520
Leerelement, 391
erzeugen bei Import, 391
Lesenzeichen, 308
navigieren, 308
Lesezeichen, 534
alle löschen, 307
einfügen und löschen, 307
in Textansicht, 113
navigieren, 308
in Textansicht, 477
Little-endian, 528
Lizenz, 535, 847, 848, 849
Lizenzierte Version, 535
Lizenzüberwachung, 845
Lizenzvereinbarung, 846, 848, 849
Lizenzvertrag, 848, 849
loading, 708
Lokales Projekt, 312
Löschen, 299
Befehl, 299
Befehl aus Symbolleiste, 504
Befehl im Kontextmenü, 511
Breakpoint - XSLT Debugger, 382
Kürzel zu Befehl, 507
Symbol aus Symbolleiste, 504
Symbolleiste, 505
Zeile (Dokument Editor), 389

M

Macros, 569
creating in Scripting Environment, 569
how to call, 571
running in application, 551
Mail, 842
Main Window, 555
in Scripting Environment, 555
Makro, 513
Hinzufügen zu Menü/Symbolleiste, 513
Schaltfläche bearbeiten, 516
Skripting, 496
MapForce 2005, 359
MAPI, 842
Markup, 388
ausblenden, 388
Gemischte anzeigen, 388
in der Authentic-Ansicht, 177
kleine anzeigen, 388
große anzeigen, 388
Markup-Symbole, 180
in der Authentic-Ansicht, 180
Markup-Symbole ausblenden, 177, 180
Maximale Zellenbreite, 523
MDAC, 391
Mehrfachlizenz, 845
Menü, 511
anpassen, 511
Ansicht, 473
Bearbeiten, 298
Befehl hinzufügen/löschen, 504
Befehle löschen, 511
Browser, 478
Datei, 286
Extras, 490
Fenster, 532
Hilfe, 534
Konvertieren, 390
Makro hinzufügen, 513
Projekt, 309
SOAP, 483
Standard/XMLSPY, 511
WSDL, 480
Message, 100
Parameter hinzufügen (WSDL), 100
zu einer Operation (WSDL) hinzufügen, 100
in XSLT/XQuery Debugger, 245
Microsoft Access, 391, 448, 449
Microsoft Office, 402
Microsoft Word, 390, 402
MIME, 520
Mixed Markup-Symbole anzeigen, 177, 180
Modules, 555
in Scripting Environment, 555
in scripting projects, 549
MS SQL Server, 369, 370
Schema Extensions, 369, 370
Schema-Einstellungen, 371

© 2005 Altova GmbH

MS Visual Source Safe, 310
MSXML, 529
MSXML.DLL, 840
MSXML3, 840
Multi-user, 518
Multiuser-Lizenz, 535

N

Nach oben, 389
Zeile verschieben (Dokument Editor), 389
Nach unten, 389
Zeile verschieben (Dokument Editor), 389
Named Schema-Beziehungen, 370
MS SQL Server Schema-Einstellungen, 370
Namespace, 840
in Grid-Ansicht einfügen, 352
Namespaces, 361
Einstellungen in der Schema Design-Ansicht, 361
in Schemas, 22
Native Oberfläche ausführen, 322
Navigieren, 45
Kürzel in Schema, 45
Nebeneinander, 523
horizontal, 532
vertikal, 532
Netzwerk, 842
Kommunikation, 842, 845
Netzwerkverkehr, 842, 845
Neu laden, 518
geänderte Dateien, 291
Neue Datei, 286
erstellen, 286
Neues XML-Dokument, 51
erstellen, 51
Nicht lateinischen Schriftsysteme, 843
Nicht-XML Dateien, 520
Node, 382
aktuellen Execution Node einblenden, 382
NOTATION Deklaration, 341
als Child in Grid-Ansicht hinzufügen, 341
in Grid-Ansicht anhängen, 338
in Grid-Ansicht einfügen, 334
Konvertieren in (Grid-Ansicht), 343

O

ODBC, 391, 448, 449
Office 2000, 391
Öffnen, 289
Projekt-Versionskontrolle, 312
OLE DB, 391
OLE DB Anbieter, 395
OLE DB Provider, 81
Online Shop, 536
Online-Hilfe, 6
für Altova-Produkte, 6
Operation, 98
an PortType anhängen, 98
Hinzufügen von Input- und Output-Messages, 100
Optimale Breite, 475, 523
Optionale Elemente, 30
erstellen, 30
Optionen, 415
von DB-Struktur, 415
Datei, 518
Oracle, 390, 391, 448, 449
Datenbank - Schema erzeugen, 410
Datenbankschema erzeugen, 81
Oracle Windows Schnittstelle, 410
Schema-Extensions, 368
Oracle XML Datenbank, 466
Oracle XML DB, 470
Datenbank durchsuchen, 470
DB durchsuchen, 466
Dokumentinstanzen hinzufügen, 471
Eigenschaften, 470
Schema auflisten, 467
Schema hinzufügen, 468
Schema öffnen, 467
Oracle XML DBUri Statements, 466
Ordner, 221
Dateien in Eclipse importieren, 221
OrgChart.sps, 88
im Tutorial der Authentic-Ansicht, 88
OrgChart.xml, 88
im Tutorial der Authentic-Ansicht, 88
OrgChart.xsd, 88
im Tutorial der Authentic-Ansicht, 88
OS, 841

© 2005 Altova GmbH

Output, 100
Message zu Operation hinzufügen, 100
Overview, 597
of XMLSpy API, 597

P

Parallellizenz, 845
Parameter, 196
in DB-Abfragen, 196
Übergabe an Stylesheet über Schnittstelle, 376
zu Message hinzufügen (WSDL), 100
Parent, 700
Parser, 838
XSLT, 529
Parser Download, 536
Patterns, 839
Perspektive, 230
XMLSpy in Eclipse, 230
Plug-In, 514
ATL sample files, 586
registration, 582
User interface configuration, 583
XMLSPY, 581
Port, 845
2799, 845
80, 842
im WSDL Editor erstellen, 98
Operation anhängen, 98
Position, 476
Zeichen, 476
Zeile, 476
Preisliste, 847
Primär-/Sekundärschlüssel, 391
ausschließen, 391
Primär-/Sekundärschlüssel - Import, 77
Primärschlüssel, 448
PRJ files, 549
for Scripting Environment, 549
Processing Instruction, 340
als Child in Grid-Ansicht hinzufügen, 340
in Grid-Ansicht anhängen, 336
in Grid-Ansicht einfügen, 332
Konvertieren in (Grid-Ansicht), 343
Programmeinstellungen, 518
Programming points, 573
in Scripting Environment, 573
Project menu, 575
in Scripting Environment, 575
Project Window, 553
in Scripting Environment, 553
Projekt, 323
aktive Dateien hinzufügen, 323
Dateien zu Projekt hinzufügen, 322
Eigenschaften, 328
entfernen aus Versionskontrolle, 319
externe Ordner hinzufügen, 323
externen Webordner hinzufügen, 326
Hinzufügen aktiver und verwandter Dateien, 323
Hinzufügen in Versionskontrolle, 317
in Eclipse definieren, 217
Menü, 309
Neu, 310
Neu laden, 310
öffnen, 310
Ordner hinzufügen, 323
schließen, 310
Skripting, 496
Skripts aktivieren, 497
Skripts entkoppeln, 497
Speichern, 310
URL zu Projekt hinzufügen, 322
zuletzt verwendete, 330
zuweisen zu Skript, 497
Versionskontrolle, 312
Projekte, 309
Übersicht, 309
erstellen, 86
Vorteile, 85
Projektfenster, 17
Projekt-Fenster, 532, 533
Projektskripts aktivieren, 497
Projektverwaltung in XMLSpy 2005, 85
Property Sheet, 557
in Form Editor, 557
Provider, 312
auswählen, 312
OLE DB und anbieterspezifisch, 81
PVCS Version Manager, 310

© 2005 Altova GmbH

R

RAM, 841
Rechschreibprüfung, 490
benutzerdefiniertes Wörterbuch, 490
Rechtliches, 846
Rechtschreibprüfung, 492
Optionen, 492
Reduzieren, 475
unselektierte reduzieren, 475
Referenz, 285
Referenz für Programmierer, 546
Register, 518
plug-in, 582
Registrieren, 468
Schema - Oracle XML DB, 468
Registrierung, 535, 842, 847
Registry, 518
Einstellungen, 518
Regular Expression, 302
in Suchstring, 302
Relationales Modell, 448
Release notes, 547
XMLSpy API, 547
Request, 484
SOAP Request senden, 484
von WSDL senden, 109
RichEdit 3.0, 525
Root, 428
Tabelle, 428
Daten importieren, 77
Rückgängig, 298
Rückgängig-Befehl, 116

S

save, 702
schema, 626, 839
Access-Datenbankschema erzeugen, 404
aus Datenbank erzeugen - Übersicht, 80
Basis für XML Import, 428
Code Generator, 262
Datenbank-Schema erzeugen, 403
DB2-Datenbankschema erzeugen, 412
DB-Struktur-Extensions, 417
Design Ansicht, 474
Dialekte, 839
Dokumentation, 47
Dokumentation Schriftart, 524
Einstellungen, 518
Einstellungen für Oracle, 369
Erstellen einer DB basierend auf Schema, 432
erzeugen von Datenbank, 81
flach SQL/XML Standard, 404
In Oracle XML DB registrieren, 468
Oracle-Datenbankschema erzeugen, 410
Schema von Datenbank öffnen, 467
siehe auch XML-Schema, 18, 354
SQL-Datenbankschema erzeugen, 407
von DB-Struktur, 415
WSDL Schema öffnen, 106
Oracle XML DB, 467
Schema Design, 361
Menü, 361
Beschreibung, 137
Diagramm anzeigen, 368
Globale Elemente anzeigen, 368
Zoom-Funktion, 367
Schema erzeugen, 417
Schema Extensions, 417
von DB-Struktur - Optionen, 415
Schema hinzufügen, 468
Oracle XML DB, 468
Schema Schriftarten, 524
Schema/WSDL Design-Ansicht, 136
Beschreibung, 136
SchemaAgent, 161, 162, 166
arbeiten mit, 161, 162, 166
erweiterte Validierung, 373
Schemas anzeigen, 373
Schemas von XMLSpy 2005 aus öffnen, 166
Verbindung trennen, 373
Verbindung zu Server herstellen von XMLSpy 2005, 373
Installation, 158
SchemaAgent in XMLSpy 2005, 155
SchemaAgent Server, 157
Installation, 157
Verbindung herstellen, 159
Schema-Ansicht, 137
Eingabehilfen in, 137
konfigurieren, 28

© 2005 Altova GmbH

SchemaDocumentationDlg, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728
AllDetails, 720
Application, 719
IncludeAttributeGroups, 720
IncludeComplexTypes, 720
IncludeEverything, 720
IncludeGlobalElements, 721
IncludeGroups, 721
IncludeIndex, 721
IncludeLocalElements, 722
IncludeRedefines, 722
IncludeSimpleTypes, 722
OptionsDialogAction, 722
OutputFile, 723
OutputFileDialogAction, 723
OutputFormat, 723
Parent, 724
ShowAnnotations, 724
ShowAttributes, 724
ShowChildren, 725
ShowConstraints, 725
ShowDiagram, 725
ShowEnumerations, 725
ShowNamespace, 726
ShowPatterns, 726
ShowProgressBar, 726
ShowProperties, 727
ShowResult, 727
ShowSingleFacets, 727
ShowSourceCode, 728
ShowType, 728
ShowUsedBy, 728
Schemas, 360
im Speicher, 360
in Schema-Ansicht erstellen, 136, 137
Schema-Übersicht, 19
ausführliche Beschreibung, 142
Grundzüge, 137
Schließen, 292
Projekt, 310
Schlüssel, 391
Primär-/Sekundärschlüssel ausschließen, 391
Schriftart, 843
Schema, 524
Schema Dokumentation, 524
Schriftgröße, 478
in Browser-Ansicht, 478
Schriftsystem, 843
Schriftsysteme, 844
Script editing, 557
in Form Editor, 557
Scripting, 497
Skript zu Projekt zuweisen, 497
documentation, 548
Form Editor, 557
Form Objects, 556
GUI description, 553
how to use, 562
launching, 553
Main Window, 555
programming points, 573
Project Window, 553
settings, 551
Scripting projects, 549
Sekundär- / Primärschlüssel, 77
Sekundär-/Primärschlüssel, 391
ausschließen, 391
Sekundärschlüssel, 448
SELECT Statement, 391
Select-Anweisung, 77
Import, 77
Sequence-Kompositor, 23
verwenden, 23
Service, 98
im WSDL Editor erstellen, 98
Shape string, 428
Automatisch erzeugen, 428
Sicherer Online Shop, 536
SimpleType, 839
in Schema-Definitionen, 34
Skripting, 496
Makros anzeigen, 496
Projekt, 496
Projektskripts aktivieren, 497
Skripts aus Projekt löschen, 497
Zu Skripting-Umgebung wechseln, 496
Skripts, 530
Skriptsprache, 530
SMTP, 842
SOAP, 488
Debugger Optionen, 488
Debugger Sitzung, 487
Menü, 483
Neuen Request erstellen, 483
Proxyserver anhalten, 488

© 2005 Altova GmbH

SOAP, 488
Proxyserver starten, 487
Request an Server senden, 484
Request Parameter, 486
Request von WSDL senden, 109
Requests, 488
Response, 488
Anwendungsbeispiel, 253
Beschreibung, 251
Breakpoints setzen, 259
in Visual Studio .NET, 214
SOAP Kommunikationsvorgang, 252
Software-Produkt, 845, 848, 849
Speicher, 310
Speichern von Schemainformationen, 360
alle, 294
Projekt, 310
SPL, 274
Splashscreen, 523
SPP Datei-Speicherpfade, 309
SPS, 286
neuer XML-Datei zuweisen, 286
zu XML-Datei zuweisen, 385
SPS-Tabelle, 90
Beispiel für statische Tabelle im Tutorial zur Authentic-Ansicht, 90
dynamischen Tabellen bearbeiten, 177
Verwendung, 188
SpyProject, 729, 730
CloseProject, 729
ProjectFile, 729
RootItems, 729
SaveProject, 730
SaveProjectAs, 730
SpyProjectItem, 730, 731, 732
ChildItems, 730
FileExtensions, 731
ItemType, 731
Name, 731
Open, 731
ParentItem, 731
Path, 732
ValidateWith, 732
XMLForXSLTransformation, 732
XSLForXMLTransformation, 732
XSLTransformationFileExtension, 732
XSLTransformationFolder, 732
SpyProjectItems, 733, 734
AddFile, 733
AddFolder, 733
AddURL, 733
Count, 734
Item, 734
RemoveItem, 734
SQL, 81
Datenbankschema erzeugen, 81
Statement für Import wählen, 77
flaches Schema, 404
SQL Server, 390, 391, 448, 449
SQL-Datenbank - Schema erzeugen, 407
Standard, 528
Encoding, 528
Menü, 511
Einstellung im Hauptfenster, 520
Standardmäßiger Editor, 520
Star Team, 310
Start, 217
Eclipse, 217
XSLT Debugger, 380
hinzufügen (Kontextmenü), 516
Statische (SPS) Tabellen in der Authentic-Ansicht, 188
Verwendung, 188
Statische Tabelle, 90
Beispiel im Tutorial zur Authentic-Ansicht, 90
Status aktualisieren, 322
Versionskontrolle, 322
Stil, 524, 525
Stoppen, 381
XSLT Debugger, 381
String, 81
connection string, 81
Strukturierter Text, 522
Studio, 206
XMLSPY Integration in Visual Studio .NET, 206
Stylesheet PI, 379, 380
StyleVision, 359
zum Bearbeiten eines StyleVision Power Stylesheet, 386
in StyleVision bearbeiten, 386
zu XML-Datei zuweisen, 385
StyleVision Power Stylesheet zu XML-Datei zuweisen, 385
Suchen, 534
Hilfe, 534
Text in Dokument, 302
und Text in Dokument ersetzen, 305
Support, 536

© 2005 Altova GmbH

Support, 536
für Altova-Produkte, 6
Support Center, 536
Symbol, 504
Hinzufügen zu Symbolleiste/Menü, 504
große anzeigen, 516
Symbolleiste, 504
Befehle anpassen, 504
Ein/Aus, 505
große Symbole anzeigen, 516
Makro hinzufügen, 513
Neue erstellen, 505
Symbolleiste & Menübefehle zurücksetzen, 505
Syntaxfärbung, 520, 523
für XQuery, 121

T

Tab sequence, 557
in Forms, 557
Tabelle, 520
automatisch erzeugen, 520
Befehl in Grid-Ansciht, 344
Farben, 526
Root Tabelle, 428
Tabelle Ansicht, 522
Tabellen, 177
dynamische (SPS)-Tabellen (SPS) bearbeiten, 177
in und aus externen Applikationen exportieren und importieren, 132
Arbeiten mit (statischen und dynamischen) SPS Tabellen, 188
Arbeiten mit XML-Tabellen, 189
Symbole zum Bearbeiten von XML-Tabellen, 193
Verwendung, 188
Tabellenansicht, 344
in Grid-Ansicht wechseln, 344
Spalten sortieren, 345
Verwendung, 64
Zeile anhängen, 345
Zeile einfügen, 345
Tabellenansicht in der Enhanced Grid-Ansicht, 132
Tamino, 372
Schema-Einstellungen, 372
Schema-Extensions, 371
Beschreibung, 451
Tastaturzuordnungen, 534
Tatstaturkürzel, 507
TCP, 845
Technische Informationen, 837
Technischer Kundendienst, 536
Technischer Support, 536
für Altova-Produkte, 6
Templates-Fenster, 245
in XSLT/XQuery Debugger, 245
terminate, 629
Text, 390
als Child in Grid-Ansicht hinzufügen, 339
Druckaufbereitung, 301
Export, 450
in Dokument suchen, 302
in Grid-Ansicht anhängen, 335
in Grid-Ansicht einfügen, 331
Konvertieren in (Grid-Ansicht), 342
Nodes ein-/ausklappen, 477
Schriftart, 525
suchen und ersetzen, 305
Textansicht, 473
Text-Ansicht, 522
Auto-Komplettierung Ein/Aus, 522
Textansicht, 55
Bearbeiten in, 55
Beschreibung, 112
Eingabehilfen, 114
intelligentes Editieren, 116
Lesezeichenleiste, 477
Text-Ansicht, 524
Schema Schriftarten, 524
TextImportExportSettings, 735, 736
DestinationFolder, 735
EnclosingCharacter, 735
Encoding, 735
EncodingByteOrder, 735
FieldDelimiter, 736
FileExtension, 736
HeaderRow, 736
ImportFile, 736
Themen, 534
Ansicht im Inhaltsverzeichnis, 534
Tiefer stufen, 346
Befehl, 346
Tooltip, 516
anzeigen, 516
Kürzel anzeigen, 516

© 2005 Altova GmbH

Training, 537
Kurse, 537
Transformation, 374
siehe XSLT-Transformation, 374
Tutorial, 17
Beispieldateien, 17
WSDL, 98
Ziele, 17
type, 53
Erweiterung im XML-Dokument, 53
integrierte, 280

U

Überlappend, 532
Fenster, 532
Übernehmen, 518
Überprüfen, 490
Rechtschreibung, 490
Übersicht, 17
Überspringen, 77
Datenbankfeld, 77
XSLT Debugger, 382
UCS-2, 528, 843
UCS-4, 843
UDP, 845
Unicode, 390, 843
Konsortium, 843
Standard, 843
Unterstützung, 843
Unicode-Zeichen, 843
Universalschriftart, 843
Unnamed Element Relationships, 371
MS SQL Server Schema-Einstellungen, 371
Unselektierte, 475
Unterschiede anzeigen, 320
Urheberrecht, 537
URL, 630, 702, 708, 842
Datei öffnen, 289
Datei speichern unter, 292
per E-Mail senden, 294
User interface, 583
configure using plug-in, 583
User Manual, 4
UTF-16, 528
UTF-8, 843

V

Validator, 838
Validieren, 351
WSDL-Datei, 104
XML-Dokument, 59
Validierung, 354
DTD zu XML-Dokument zuweisen, 354
von verknüpften Schemas mit Hilfe von SchemaAgent, 166
XML-Schema zu XML-Dokument zuweisen, 354
Validierungsparser, 838
Variablenfenster, 243
in XSLT/XQuery Debugger, 243
Verbergen, 532, 533
Verbindung mit SchemaAgent Server trennen, 373
Verbindung zu SchemaAgent Server herstellen, 159
Verbindung zu SchemaAgent Server von XMLSpy 2005, 373
Vergrößern/Verkleinern, 367
in Schema Design-Ansicht, 367
Verknüpfen, 98
Adresse mit einem Binding (WSDL), 98
Verlauf anzeigen, 319
Version changes, 547
XMLSpy API, 547
Versionsdateien vergleichen, 320
Versionskontrolle, 310
aktivieren, 314
aktuellste Version holen, 315
auschecken, 315
Auschecken rückgängig, 316
ausgliedern, 319
deaktivieren, 314
Eigenschaften, 321
Einchecken, 316
Hinzufügen in Versionskontrolle, 317
native Oberfläche ausführen, 322
Projekt öffnen, 312
Status aktualisieren, 322
Unterschiede anzeigen, 320
Verlauf anzeigen, 319
Versionsnummer, 537
Vertrag, 846
Vertrieb, 847
Verwandte Daten, 390

© 2005 Altova GmbH

Verwendung, 845, 849
Lizenz, 845, 849
Verzeichnisse, 499
zwei vergleichen, 499
Verzeichnisse vergleichen, 499
View menu, 575
in Scripting Environment, 575
Visual Studio .Net, 214
und SOAP Debugger, 214
und XMLSPY, 206
und XMLSPY Unterschiede, 207
und XSLT Debugger, 211
Volltextsuche, 534
Von rechts nach links, 844
Vorlagen, 17
Ordner, 17
von XML-Dokumenten in Authentic-Ansicht, 384
für neue Dokumente, 286
Vorschau, 390

W

W3C, 838, 839
Web Server, 536, 537
Web Shop, 536
Webservice, 106
Verbindung herstellen, 106
Wechseln, 496
Zu Skript-Umgebung, 496
Weitersuchen, 305
Befehl, 305
Whitespace-Behandlung, 538
in XSLT Transformationen, 538
in XSLT-Transformationen, 539
Wiedergabe, 844
Wiederherstellen, 298
Befehl, 298
Wiederholende Elemente, 522
Window menu, 580
in Scripting Environment, 580
Windows 2000, 391, 841, 843
Windows 95, 841
Windows 98, 841
Windows NT, 841, 843
Wohlgeformtheitsprüfung, 350
bei XML-Dokument, 59
Word, 390, 402
Word Dokument, 402
Workbench, 217
Eclipse, 217
Wörterbuch, 490
aktuelles ändern, 490
benutzerdefiniertes Wörterbuch hinzufügen, 490
Rechschreibprüfung, 490
WSDL, 98
Adresse mit einem Binding verknüpfen, 98
Binding, 481
Binding-Typ definieren, 100
Dokumentation erstellen, 110
Einstellungen, 482
Erzeugen von Dokumentation, 482
Messages, 480
Operations, 480
PortType, 480
Schema öffnen, 106
Service, 481
SOAP Debugger-Beispiel, 253
SOAP Request senden, 109
Tutorial, 98
Types, 481
validieren, 104
Verbindung zu einem Webservice herstellen, 106
Web service, 483
Beschreibung, 169
GUI-Komponenten, 171
WSDL Menü, 480
WSDL-Dokumente, 169
Editierschnittstelle in XMLSpy 2005, 169

X

XDR, 838, 839
XML, 299, 402, 843
Dokument, 50
Hinzufügen von Dokumentinstanzen in Datenbank, 471
in Eclipse erstellen, 219
in Eclipse importieren, 221
Menü, 331
Neues Dokument, 51
Parser, 838
Rechschreibprüfung, 490
Symbole, 50

© 2005 Altova GmbH

XML, 299, 402, 843
Dateien vergleichen, 501
Dateivergleich, 497
Verzeichnisse vergleichen, 499
XML Dokument, 470
Oracle XML DB durchsuchen, 470
XML Import, 428
Basierend auf Schema, 428
XML Schema, 839
MS SQL Server Schema-Einstellungen, 371
XMLData, 737, 738, 739, 740, 741, 742, 743
XML-Data, 839
XMLData, 737
AppendChild, 737
EraseAllChildren, 738
EraseCurrentChild, 738
GetCurrentChild, 739
GetFirstChild, 739
GetNextChild, 740
HasChildren, 741
InsertChild, 741
IsSameNode, 742
Kind, 742
MayHaveChildren, 742
Name, 742
Parent, 743
TextValue, 743
XML-Datei, 359
von DTD oder XML-Schema erzeugen, 359
XML-Dokument, 88
in der Authentic-Ansicht öffnen, 88
in Textansicht bearbeiten, 55
to XSLT Stylesheet zuweisen, 380
XML-Format, 390
XMLINST.EXE, 840
XML-Konformität, 520
XML-Prolog, 340
als Child in Grid-Ansicht hinzufügen, 340
in Grid-Ansicht anhängen, 336
in Grid-Ansicht einfügen, 332
Konvertieren in (Grid-Ansicht), 343
XML-Schema, 838
Ansicht konfigurieren, 28
Code generieren von, 360
Content Model-Ansicht konfigurieren, 364
Content Model-Diagramm, 362
Dokumentation generieren, 362
Einstellungen in der Schema Design-Ansicht, 361
Erstellen eines einfachen Schemas, 18
Gehe von XML-Dokument zu Definition, 355
Gehe von XML-Dokument zu Schema, 355
in DTD konvertieren, 356
Komponenten hinzufügen, 23
Menübefehle zu, 354
MS SQL Server Extensions, 369
MS SQL Server Schema-Einstellungen, 370
Namespace-Einstellungen in der Schema Design-Ansicht, 361
Namespaces definieren, 22
Navigation in der Design-Ansicht, 45
neues erstellen, 19
Oracle Extensions, 368
Oracle Schema-Einstellungen, 369
siehe auch Schema, 354
Tamino Extensions, 371
Tamino Schema-Einstellungen, 372
Tutorial, 18
von XML-Dokument generieren, 355
während der Bearbeitung des XML-Dokuments ändern, 68
XML-Datei erzeugen, 359
zu XML-Dokument zuweisen, 354
XMLSPY, 285, 537
allgemeine Einführung, 5
Eclipse-Ansichten und Perspektiven, 230
plug-in registration, 582
Training, 537
und Eclipse, 216
und Eclipse SOAP Debugger, 227
und Eclipse XSLT Debugger, 225
Unterschiede zu Eclipse, 224
documentation, 596
overview, 597
release notes, 547
XMLSpy Enterprise Edition, 1
Benutzerhandbuch, 1
XMLSPY plug-in, 581
XMLSpyDocumentEditor, 782
MarkUpView, 782
XMLSpyLib, 596, 617, 632, 633, 660, 675, 679, 682, 684, 706, 709, 710, 711, 713, 715, 718, 729, 730, 733, 734, 736, 744, 760, 763
Application, 617
AuthenticDataTransfer, 632
AuthenticRange, 633
AuthenticSelection (obsolete), 760
AuthenticView, 660

© 2005 Altova GmbH

XMLSpyLib, 596, 617, 632, 633, 660, 675, 679, 682, 684, 706, 709, 710, 711, 713, 715, 718, 729, 730, 733, 734, 736, 744, 760, 763
CodeGeneratorDlg, 675
DatabaseConnection, 679
Dialogs, 682
DocEditEvent (obsolete), 744
DocEditView (obsolete), 763
Document, 684
Documents, 706
ElementList, 709
ElementListItem, 710
ExportSettings, 711
FileSelectionDlg, 713
GridView, 715
ProjectItem, 730
SchemaDocumentationDlg, 718
SpyProject, 729
SpyProjectItems, 733
TextImportExportSettings, 734
XMLData, 736
XML-Tabellen in der Authentic-Ansicht, 193
Editiersymbole, 193
Verwendung, 189
XML-Text, 522
kopieren, 299
XML-Vorlagendatei, 88
in der Authentic-Ansicht, 88
XPath 1.0, 347
in XPath Evaluator, 347
XPath 2.0, 347
in XPath Evaluator, 347
XPath Evaluator, 347
Verwendung, 347
XPath kopieren, 301
XPath-Watch-Fenster, 244
in XSLT/XQuery Debugger, 244
XQuery, 542
Altova XQuery Prozessor, 542
Ausführung, 124
Datei öffnen, 119
Dokumentenvalidierung, 124
Eingabehilfen, 120
in Textansicht bearbeiten, 119
intelligente Editierfeatures, 123
Syntaxfärbung, 121
Variablen an XQuery-Dokument übergeben, 376
siehe XSLT/XQuery Debugger, 232
XQuery Prozessor, 542
Informationen, 542
XQuery-Ausführung, 379
XQuery-Dateien, 119
Dateierweiterungen in XMLSpy 2005 festlegen, 119
XSD, 838, 839
xsi:type, 53
Verwendung, 53
XSL, 380
siehe XSLT, 380
XSL Stylesheet, 840
XSL Transformation, 840
XSL/Query Menü, 374
XSL:FO, 374
Transformation, 374
XSLT, 476, 840
Altova XSLT 1.0 Prozessor, 538
Altova XSLT 2.0 Prozessor, 539
in XMLSpy 2005 ändern, 73
Prozessor, 529
Informationen, 538
XSLT 2.0 Prozessor, 539
Informationen, 539
XSLT Debugger, 382
aktuelle Execution Nodes einblenden, 382
aussteigen, 381
Breakpoint einfügen/löschen, 382
Breakpoints Dialogfeld, 382
Breakpoints Ein/Aus, 382
Debug-Fenster, 383
Debugger erneut starten, 381
Debugger starten, 380
Debugger stoppen, 381
Debuggersitzung beenden, 381
einsteigen, 381
Einstellungen, 383
in Visual Studio .NET, 211
überspringen, 382
XSLT Stylesheet, 380
öffnen, 380
zu XML-Dokument zuweisen, 379, 380
zu XML-Dokument zuweisen, 380
XSLT TransFormation System, 536
Download, 536
XSLT Transformationen, 448
XSLT/XQuery debuggen, 232
verwendete Dateien, 232

© 2005 Altova GmbH

XSLT/XQuery Debugger, 233
Beschreibung der Benutzeroberfläche, 233
Beschreibung der Funktionsweise, 233
Einstellungen, 238
Fenster "Call Stack", 244
Fenster "Info", 246
Fenster "Messages", 245
Fenster "Templates", 245
Fenster "Variablen", 243
Fenster "XPath-Watch", 244
Funktionen und Verwendung, 232
Informationsfenster, 242
Kontextfenster, 243
Sitzung starten, 240
toolbar icons, 235
Verwendung von Breakpoints, 248
XSLT-Parameter, 376
über Schnittstelle an Stylesheet übergeben, 376
XSLT-Prozessor, 840
XSL-Transformation, 70
siehe XSLT, 70
XSLT-Transformation, 374
in FO, 374
in PDF, 374
in XMLSpy 2005, 72
Tutorial, 70
XSLT-Datei zuweisen, 71

Z

Zeichen, 476
Position, 476
Zeichensatz, 390
Kodierung, 528
Zeile, 388
anhängen (Dokument Editor), 388
einfügen (Dokument Editor), 388
Gehe zu, 476
kopieren (Dokument Editor), 388
löschen (Dokument Editor), 389
nach oben verschieben, 389
nach unten (Dokument Editor), 389
anhängen, 345
einfügen, 345
Zeilenlänge, 475
Zeilenumbruch in Textansicht, 475
Zeilennummerierung, 113
in Textansicht, 113
Zeilenschaltung, 90
Auswirkung, 90
siehe Eingabetaste, 204
Zeilenumbruch, 518
aktivieren/dekativiere, 475
Ein/Aus, 475
Zuletzt verwendetet Dateien, 297
Liste, 297
Zurücksetzen, 507
Kürzel, 507
Menübefehle, 511
Symbolleiste & Menübefehle, 505
Zuweisen, 507
Kürzel zu Befehl, 507
Skript zu Projekt, 497

© 2005 Altova GmbH